



Overlay Network Mechanisms for Peer-to-Peer Systems

Vom Fachbereich
Informatik
der Technischen Universität Darmstadt
genehmigte
Dissertationsschrift
in englischer Sprache

von

M.A.Sc. Vasilios Darlagiannis

geboren am 08.06.1973 in Kozani, Griechenland

zur Erlangung des Grades eines
Doktor-Ingenieur (Dr.-Ing.)

Darmstadt 2005
Hochschulkennziffer D17

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Nicolas D. Georganas (Ph.D.)

Tag der Einreichung: 29.04.2005
Tag der Disputation: 14.06.2005

Kurzfassung (Deutsch)

Das *Peer-to-Peer* (P2P) Paradigma bietet einen alternativen Designansatz für Verteilte Systeme, das die Forderung nach einem dedizierten Dienstanbieter, wie im *Client-Server* Fall notwendig, lockert. Der P2P-Ansatz ergründet das Potential Verteilte Systeme zu schaffen, die auf Ressourcen und Diensten beruhen, die durch jedes End-Gerät, das an ein herkömmliches Kommunikationsnetz wie das Internet angeschlossen ist, zur Verfügung gestellt werden. Verteilte Systeme basierend auf dem P2P-Paradigma formen, virtuelle Netzwerke, so genannte *Overlay-Netzwerke*, die auf physikalischen Telekommunikationsnetzwerken aufsetzen. Ein Overlay-Netzwerk ist hierbei eine notwendige Abstraktion, die sowohl die funktionalen Anforderungen, wie z. B. Verbindungsverwaltung, Indizierung und Routing, erfüllen muss, als auch nicht-funktionale Eigenschaften, wie z. B. Skalierbarkeit, Ausfallsicherheit und Lastausgleich, berücksichtigen sollte. Eine Herausforderung ist hierbei Overlay-Netzwerke so zu gestalten, dass die gestellten Anforderungen befriedigt werden und gleichzeitig potentielle Konflikte zwischen diesen Anforderungen auf geeignete Weise gelöst werden.

Diese Dissertationsschrift erforscht eine neuartige Architektur zur Entwicklung von effizienten Overlay-Netzwerken für P2P-Systeme, welche die wichtigsten Charakteristiken von häufig eingesetzten Systemen berücksichtigt. Im speziellen wird die Anzahl von Teilnehmern berücksichtigt, die zu *Skalierbarkeits-* und *Erweiterbarkeits-*Problemen führen kann. Darüber hinaus befasst sich die Arbeit mit dem *unkontrollierbaren* und schwer vorherzusehenden Verhalten der Knoten von P2P-Systemen, was zu hoch dynamischen, schwankenden und veränderlichen Overlay-Netzwerk-Topologien führt. Durch dieses Verhalten wird der Entwurf *ausfallsicherer* und *stabiler* Systeme, die *effizient* und mit minimalen Wartungskosten betrieben werden können, zur Herausforderung. Zusätzlich muss auch die intrinsische *Heterogenität* der physischen Eigenschaften und das Benutzerverhalten berücksichtigt werden, die eine *gerechte Verteilung* der Aufgaben erschwert.

Um die oben aufgeführten Anforderungen zu erfüllen, wurde eine Anzahl von Mechanismen entworfen. Für die Netzwerktopologie wird die Nutzung von *de Bruijn*-Graphen vorgeschlagen. Sie besitzen attraktive Eigenschaften, da ihr Graph-Durchmesser logarithmisch zur Grösse des Graphen wächst, sogar bei konstanter Kantenzahl. Dies ist besonders nützlich, um den Anforderungen an Skalierbarkeit mit minimalen Wartungskosten gerecht zu werden. Jedoch erweist sich hierbei die nur exponentielle Erweiterbarkeit von *de Bruijn*-Graphen als Problem. Ein Algorithmus zum Aufbau von Graphen basierend auf lokaler Information wurde vorgeschlagen, um *de Bruijn*-Varianten mit inkrementellen Erweiterungseigenschaften zu definieren. Darüber hinaus gibt es aber auch Bedenken bzgl. der Ausfallsicherheit wegen der konstante Anzahl von Kanten pro Knoten, die vorzugsweise

so klein wie möglich sein sollte, um die Wartungskosten des Graphen gering zu halten. Dieses Problem wird durch die Einführung von *Peer-Cluster*, durch die Zuverlässigkeit garantiert wird, und eine so genannte *Two-Tier-Topologie*, bei der die de Bruijn-Struktur auf die Inter-Cluster-Verbindungen angewandt wird, adressiert. Durch diese *hybride* Topologie ergibt sich ein *wohl strukturiertes* Netzwerk. Parallel hierzu erlaubt dies die Wahl direkter Nachbar-Peers unter mehreren Peers eines Nachbar-Clusters. Dies wird durch unterschiedliche Richtlinien und Metriken gesteuert, d.h. einer effizienten Abbildung auf die unterliegende Netzwerkstruktur oder durch so genannte Trust-Level-Richtlinien. Ferner werden Mechanismen für die Intra-Cluster-Organisation vorgeschlagen, die die Heterogenität der einzelnen Peers einbeziehen. Hierfür wurde ein *Rollen*-basierter Ansatz entwickelt. Die allgemeinen Kernoperationen von P2P-Overlay-Netzwerken wurden identifiziert, wobei jeder Operation eine Rolle zugeordnet wurde. Vier Basis-Rollen wurden identifiziert: *Routers*, *Cachers*, *Indexers* und *Maintainers*. Peers werden Rollen aufgrund ihrer Fähigkeiten und vorhergesagtem Verhalten zugewiesen, damit jeder Peer effizient und ohne die Gesamtleistung des Systems zu behindern oder zu verringern zum System beiträgt. Zuverlässige Regeln sorgen hierbei für ausgeglichene Beiträge eines jedes Peers, was zu einer gerechten Lösung führt. Die vorgeschlagene Architektur wurde im *Omicron*-Ansatz umgesetzt. Das resultierende System wurde sowohl analytisch als auch durch Simulation evaluiert.

Der analytische Ansatz basiert auf stochastischer Modellierung sowie der Analyse und Evaluation der Mechanismen, die Daten von empirischen Untersuchungen breit eingesetzter P2P Systemen berücksichtigt. Des weiteren werden so genannte "burn-in" Methoden, die verhältnisbezogene Zuverlässigkeitsbetrachtungsweisen nutzen, zur Optimierung der Rollenzuordnung eingesetzt.

Für die Simulation wurde ein Werkzeug benötigt, das in der Lage ist, die Dynamik des Benutzerverhalten, die Eigenschaften des unterliegenden Netzwerks und das Zusammenspiel der eingebundenen P2P-Protokolle abzubilden. Für die Evaluation der Arbeit wurde ein solches Werkzeug basierend auf einer offenen Architektur entwickelt, das ebenfalls andere wichtige Ansätze einschliesst. Damit werden vergleichende quantitative Leistungsmessungen für eine Vielzahl verschiedener interessanter Einsatzszenarien ermöglicht.

Sowohl die Ergebnisse der Analyse als auch der Simulation bestätigen das überlegene Design von Omicron für grosse, dynamische und heterogene Umgebungen, und seine Fähigkeit effizient die grosse Zahl von Zielkonflikten der Anforderungsbeschränkungen aufzulösen.

Abstract

The *Peer-to-Peer* (P2P) paradigm provides an alternative design approach for distributed systems, which relaxes the requirement for dedicated service providers as it is the case in *client-server* systems. The P2P approach explores the potential to create distributed systems based on the resources and the services that can be provided by any end-point device connected to a common communication medium, i.e. the Internet. P2P-based distributed systems develop dedicated virtual networks on top of physical telecommunication networks, the so-called *overlay networks*. An overlay network is a mandatory abstraction from physical networks to both flexibly fulfill functional requirements such as connectivity maintenance, indexing and routing, as well as to satisfy non-functional ones, such as scalability, fault-tolerance and load-balance. A challenging aspect in designing overlay networks is the satisfaction of the posed requirements, while coping with potential conflicts among them in a customizable way.

This thesis investigates a novel architecture for designing effective overlay networks for P2P systems that considers the most important characteristics of widely deployed systems. In particular, the large number of participants is considered, which introduces network *scalability* and *expandability* issues. Moreover, the thesis is concerned with their *uncontrollable* and difficult to predict behavior, which causes highly dynamic and fluctuating overlay network topologies. Such behavior makes it challenging to design *resilient* and *stable* systems that can operate *effectively* with minimum maintenance cost. Additionally, an important consideration is the intrinsic *heterogeneity* of their physical capabilities and user behavior that aggravates the problem of *even workload distribution*.

A number of mechanisms have been devised to meet the aforementioned requirements. Starting with the network topology, the usage of *de Bruijn* graphs is proposed. Their attractive characteristic of having a logarithmically increasing diameter, even when nodes have fixed degree, is particularly useful to address the scalability requirement with minimum maintenance cost. However, the exponential expandability is an intrinsic issue for de Bruijn graphs. A graph construction algorithm based only on local information has been proposed to define de Bruijn variants with incremental expandability properties. Moreover, the fixed node degree (preferably as small as possible to keep the graph maintenance costs low) poses resiliency concerns. This matter is addressed with the introduction of peer *clusters* that guarantee their reliability and with a *two-tier topology* where the de Bruijn structure applies at the inter-cluster connections. This *hybrid* topology provides a *tightly structured* network. In parallel, it gives the freedom of selecting neighbor peers from several members of the neighbor clusters. This selection can be driven by various policies and metrics, i.e., by the efficient mapping to the underlying network or by satisfying trust-level

requirements. Additional mechanisms have been proposed for the intra-cluster organization that deals with peer heterogeneity. For this issue, a *role*-based approach has been investigated. The common, basic operations of P2P overlay networks have been identified assigning a role to each of them. More specifically, four core roles have been identified: *Routers, Cachers, Indexers and Maintainers*. Peers are assigned with roles based on their capabilities and their predicted behavior so that each peer can contribute in an efficient way without hindering and degrading the overall performance. Certain rules increase the balanced contribution of each peer resulting in a fair solution. The proposed architecture has been realized within the *Omicron* approach. The resulting system has been evaluated both analytically and by simulation.

The analytical approach is based on stochastic modeling, analysis and evaluation of the mechanisms that take into account empirically observed measurements collected from widely deployed P2P systems. Further, burn-in methods that capitalize on conditional reliability approaches are utilized in order to provide an optimal role assignment decision.

The simulative approach required the usage of a tool capable of accurately capturing the dynamics of user behavior, the characteristics of the underlying networks and the detailed interaction of the involved P2P protocols. An open architecture simulation framework has been developed to augment the evaluation of our work as well as other important alternative solutions. Thereby, comparative quantitative measures of the performance in a wide range of interesting scenarios are provided.

Both the analytical and the simulation results demonstrate the superior design of Omicron for large scale, dynamic and heterogeneous environments as well as its ability to effectively adapt a plethora of trade-offs in requirement constraint.

Acknowledgments

KOM has been my home for the last five years. Professor Steinmetz has had the demanding role of directing me in this open-end endeavor and of providing his prudent guidance to help me achieving my goal. Admittedly, I feel very fortunate having such an excellent mentor. I would like to thank him sincerely for providing the environment for my research and work in all this time. Also, my co-adviser Professor Georganas was an ideal supervisor from my Master's degree onwards and still has been offering his continuous support to me. I am indebted to both of them.

I was amazingly lucky to have Dr. $\alpha\sigma\epsilon\iota\beta\eta\Lambda$ Mauthe as my group head and direct supervisor. $\alpha\sigma\epsilon\iota\beta\eta\Lambda$ (certainly a da Vinci fan) who managed to inspire me while keeping the intricate balance between the desired freedom and the necessary guidance. $\alpha\sigma\epsilon\iota\beta\eta\Lambda$ unconditionally offered his support even after his departure from our institution. In addition, I am grateful to Dr. Martin Karsten (Kalli) who had ' $\alpha\sigma\epsilon\iota\beta\eta\Lambda$ ' role during my first years at KOM. Kalli showed me the hard path I had to follow to achieve my goals.

However, even before my arrival at Germany, Abed offered me his endless hospitality and a position inside his family. He has always been a real friend. Similarly, I felt immediately very close to Ivica. Admittedly, Professor Steinmetz puts a lot of effort into putting together appropriate teams. I was very lucky for always having excellent partners -Nico, Oliver and Ralf- who proved again and again that they are very valuable friends, too. Having such friends, makes work fun! In addition, I am grateful to our partners in the MMAPPS project, in particular to Burkhard, Costas, Jan and David.

Further, I would like to express my gratitude to Matthias who has always provided wise comments. Similarly, Manuel, the "Latex guru", has always been available for sharing his knowledge. Moreover, Jens, Utz, Michael, Giwon, Rainer, Ivan, Krishna, Luka, Tronje, Alejandro, Hannes, Lipi, Amir, Carsten, Andreas, Stefan, Parag, Cornelia and Christoph have been ideal colleagues. I wish I will have the opportunity to share again my goals with them. Of course, I shall not forget the kind behavior of Ms. Kolb, Heike, Jan, Frank, Monika, Sabine, Karola and Ms. Ehlhardt who provided a smooth environment to work in.

Also, I would like to thank my personal friends here in Darmstadt: Stefanos, Dimitris, Nasos, Christos and Filippas. Without them, I would have been more lonely.

Finally and most importantly, I owe all my accomplishments to the endless support offered by my family and Tasa. It has been almost eleven years now!

Στην Τάσα.

Contents

Kurzfassung	ii
Abstract	v
Acknowledgments	vii
Preface	xiii
1. Introduction	1
1.1. Overview	2
1.2. Motivation	3
1.3. Problem Statement and Goal	5
1.4. Approach	5
1.5. Contribution	7
1.6. Outline	9
2. Overlay Network Requirements	11
2.1. Motivation	12
2.2. Applications and Services	12
2.2.1. File Sharing Applications and Services	12
2.2.2. Beyond File Sharing	14
2.3. Graph Theory and Networks	17
2.3.1. Small-world Networks	18
2.3.2. Scale-free Networks	19
2.3.3. Lexicographic Digraphs	19
2.4. Overlay Functionality	20
2.4.1. User-triggered Operations	21
2.4.2. System-triggered Operations	22
2.5. Non-functional Requirements	23
2.5.1. Scalability	23
2.5.2. Expandability	24
2.5.3. Dependability	24
2.5.4. Load-balance, Fairness and Heterogeneity	27
2.5.5. Efficiency	28
2.5.6. Autonomy	29
2.5.7. Anonymity	29

2.6. Constraint Requirements and Trade-offs	29
2.7. Summary	30
3. Alternative Overlay Network Design Approaches	33
3.1. Overlay Network Design Dimensions	34
3.2. Distributed Hash Tables	36
3.2.1. Chord	37
3.2.2. de Bruijn Digraph-based DHTs	38
3.2.3. Further Approaches	39
3.3. Hybrid Topologies	40
3.3.1. JXTA	40
3.3.2. Brocade	41
3.3.3. SHARK	43
3.3.4. OceanStore	44
3.3.5. Hybrid PIER	45
3.3.6. Further Hybrid Mechanisms	45
3.4. Summary	45
4. Overlay Network Architecture	47
4.1. Motivation	48
4.2. Core Components	49
4.2.1. Common Components	49
4.2.2. Supplementary Components	50
4.3. Basic Mechanisms	50
4.3.1. Clustering	51
4.3.2. Entity Identification Scheme	53
4.3.3. Roles	53
4.3.4. Incentives and Rules	55
4.3.5. Two-tier Network Architecture	56
4.4. Omicron Architecture	57
4.5. Overlay Network Management	58
4.5.1. Join Operation	58
4.5.2. Expandability	60
4.5.3. Overlay Stabilization	62
4.5.4. Enhanced de Bruijn Topology	64
4.6. Summary	64
5. Protocols and Role Mechanisms	67
5.1. Motivation	68
5.2. Protocol Basics and Inter-peer Communication	68
5.2.1. Common Protocol Functionality	68
5.2.2. Inter-role Communication Model	70
5.2.3. Connector	70
5.3. Router	72
5.3.1. Routing Algorithms	73
5.3.2. Messages	74

5.4. Maintainer	75
5.4.1. Functionality	75
5.4.2. Messages	76
5.5. Indexer	77
5.5.1. Functionality	77
5.5.2. Messages	77
5.6. Cacher	78
5.6.1. Motivation	78
5.6.2. Index Caching Mechanism Design	80
5.6.3. Algorithms	82
5.7. Inter-role Scenarios	83
5.7.1. Lookup Request	83
5.7.2. Resource Advertisement	85
5.7.3. Join Request	85
5.8. Summary	87
6. Optimal Role Assignment	89
6.1. Motivation	90
6.2. Distribution Generative Models	91
6.3. Empirical Observations	92
6.3.1. Measurements Analysis	92
6.3.2. Critical Factors	95
6.3.3. Empirical Fitting	95
6.4. Modeling	97
6.4.1. Peer Life-cycle Model	97
6.4.2. Swarm-based File Sharing Model	99
6.4.3. Peer Role Cost Model	101
6.5. Burn-in Optimization	103
6.5.1. Conditions	103
6.5.2. Criteria	104
6.5.3. Mission Time Estimation	105
6.5.4. Cost-based Optimization	106
6.5.5. An Example Use Case	106
6.6. Cluster endurance	108
6.7. Summary	109
7. Open Architecture Simulation Framework	111
7.1. Motivation	112
7.2. Simulator Design	113
7.2.1. Functionality Layers	113
7.2.2. Components	114
7.2.3. Simulation Engine	117
7.3. Underlying Network Support	117
7.4. Roles	119
7.5. Message Handling	120
7.6. Summary	121

8. Evaluation Experiments	123
8.1. Experiments Description	124
8.2. Scalability	124
8.2.1. Impact on Shortest Path Distribution	124
8.2.2. Impact on Average Query Length	126
8.3. Network Proximity	127
8.4. Routing Workload Distribution	129
8.4.1. Query Workload Distribution	129
8.4.2. Workload Distribution and Efficiency	131
8.4.3. Enhanced de Bruijn Topology	131
8.5. Maintenance Overhead and Accomplished Cluster Coverage	133
8.5.1. Investigated Algorithms	133
8.5.2. Cluster Coverage	136
8.5.3. Cluster Sampling Inter-arrival Distribution	138
8.6. Indices Caching Mechanism Performance	140
8.7. Summary	143
9. Conclusions and Outlook	145
9.1. Summary	146
9.2. Conclusions	147
9.3. Outlook	148
Bibliography	149
Author's Publications	167
Curriculum Vitae	171
A. Discrete and Continuous Distributions	175
A.1. Weibull distribution	175
A.2. Power law and Pareto distribution	175
A.3. Lognormal distribution	175
B. Porting Chord Algorithms into the Open Architecture P2P Simulator	177
C. Basic Concepts of Collaborative Virtual Environments	181
D. Additional Simulative Results	183
E. Simulator Performance Evaluation	185
F. Abbreviations	187
List of Figures	189
List of Tables	191

Preface

Humans spend considerable amount of time *looking for* new information or items they vaguely remember their latest location. A particularly related factor may be the mentality of each individual, which is certainly affected by a number of factors, e.g. environment, personality, etc. For example, we (the Greeks) are not considerably *organized* (probably the sea and the sun distract and enchant our minds) resulting in spending a substantial amount of time repeating actions we could have avoided if our environment was more *structured*. On the other hand, considering my personal experience in Germany, I could safely conclude that the disciplined life-style is a major factor for the achieved *productivity*.

While it is very hard to change human mentality (otherwise education, religion and governments' policies would have succeeded it) which is unethical and immoral in many cases, it is a lot simpler to develop a perfectly structured and organized *virtual cybernetic* environment that follows without distractions their creators' directions. Such a strategy may be a promising alternative way to improve productivity, which is apparently what our science particularly focuses on.

The way to accomplish the aforementioned goal may differ, based on several considerable parameters, though. However, *finance* and *profitable investment* proved so far to be the most significant factors driving the opted strategy for widely deployed solutions, since companies have to be financially viable. Thus, we have witnessed an enormous development of database and Web related technologies based on the *Client/Server* paradigm. Colossal infrastructures dominate today's technological services map. A momentous example of such an approach are the search services offered by companies such as Google. Personally, my everyday life is dependent on the availability of these services. Admittedly, in most of the cases I am very satisfied with the provided services.

However, in the last few years, we have also witnessed a parallel effort mostly driven by SMEs (small and medium enterprisers), individuals and universities aiming to harness the power of the so-called *disruptive* computing technologies. The latter approach is mostly based on the *Peer-to-Peer* paradigm resulting in a *modern era* of Peer-to-Peer systems. Such efforts are mainly driven either by factors other than purely financial or by the need to introduce more competing models relating profit and offered services. For example, we may refer to the tremendous success of Skype, a widely used telephony service.

A great debate may raise on whether Peer-to-Peer, Client/Server or hybrid approaches will eventually dominate. In fact, a number of additional factors such as *anti-censorship*, *security*, *performance*, *scalability*, *fault-tolerance* and *mobility* will play a role in deciding about success or failure of these concepts.

Contents

However, from a scientific point of view, pure Peer-to-Peer and hybrid approaches are more appealing, mostly because they are less explored and because of their attractive increased complexity. In the context of this work, we have spent considerable efforts in order to provide a piece of contribution in designing mechanisms for hybrid Peer-to-Peer systems by addressing the raised requirements found in modern Peer-to-Peer applications.

Vasilios Darlagiannis
Darmstadt, 26th of March 2005

Introduction

Tell me, O Muse, of that creative man
who traveled far and wide...

HOMER, BEGINNING OF ODYSSEY

Conceptual outline.

This chapter introduces and motivates the challenging aspects of this thesis in the area of Peer-to-Peer systems. Initially, the conceptual context of this work is defined in terms of widely deployed distributed systems by principally elaborating the differences between the Client/Server and Peer-to-Peer communication paradigms. This discussion leads to the identification of the conceptual borders of our work area, that of designing effective overlay networks under multiple conflicting requirements and constraints raised by considering a large-scale, dynamic and heterogeneous environment, thereby, reflecting the present environmental conditions of Internet-scale P2P applications. We then detail the goals and the contributions of this thesis particularly in developing overlay networks and provide efficient communication means for Peer-to-Peer systems. The specific approach used to reach the goals and the related assumptions are also reviewed, followed by the road-map of this dissertation.

This chapter is organized as follows. We provide an introductory overview of distributed systems in Section 1.1. The importance of the problem tackled in this thesis is discussed in Section 1.2 where the need for deeper investigation is motivated. Afterwards, the problem statement and the major goals of this thesis are described in Section 1.3 followed by the selected approach in Section 1.4 and the achieved scientific contributions in Section 1.5. Finally, the chapter is completed by briefly providing the outline of this dissertation in Section 1.6.

1.1. Overview

The need of everyday life to access vast amounts of diverse and constantly updated information and services limits significantly the utility gained by stand-alone systems. This inevitable limitation boosts the universal usage of distributed systems, mostly connected via the Internet. In fact, Internet-based distributed systems have been evolved to essential information sources for our society.

A very general definition of distributed systems is provided in [AW04]: *"A distributed system is a collection of individual computing devices that can communicate with each other."* Such a general definition encompasses a wide range of computer systems. However, the focus of this dissertation is on distributed systems composed of loosely coupled, autonomous entities. In this context, each entity has its own semi-independent agenda, though, limited coordination and cooperation of these entities are mandatory to meet their individual goals, i.e., sharing of information, services and resources.

Typically, distributed system architectures follow two basic paradigms: (i) Client/Server (C/S) [Sin92], [Adl95], [EM94], or (ii) Peer-to-Peer (P2P) [SW05], [SW04b], [ATS04], [Sch01], [MH03]. In the C/S paradigm, participants are assigned to unambiguous roles of either service consumers (clients) or service providers (servers). However, the participating entities can have different roles in cascading subsystems, e.g., an entity that acts like a server in the first level can act as a client in a following level. On the other hand, the entities of P2P systems have potentially equivalent responsibilities and they may concurrently act both as service consumers and providers. Hybrids of the aforementioned paradigms have been also successfully exploited, i.e., certain services of the system are deployed following the C/S approach, while other services are built adapting the P2P approach (e.g., Napster [Nap05]).

The C/S paradigm is the most widely deployed approach in Internet services today. The most crucial factors for its deployment are (i) the relative low complexity in designing a centralized solution as compared to a highly distributed one, (ii) the controlled operation of the services by usually trusted vendors, (iii) the intrinsic matching with efficient business models allowing providers to profit from the service deployment, and (iv) the technological achievements of the past and the higher requirements of providing services rather than consuming them. These factors reasonably led to the decision of acquiring fewer costly and complex computers that can be shared by many low cost end-point devices.

However, because of the intrinsic limitations of the C/S paradigm as well as the latest technological achievements and the wide acceptance of the Internet as a communication medium, a change towards systems that adopt the P2P or hybrid paradigms can be observed in the last years. Apparently, the centralized nature of the C/S paradigm makes it an easier target for distributed Denial of Service (DoS) attacks, as opposed to the P2P alternative. Moreover, the limited physical capabilities of each server poses restrictions on the number of clients that can be served, making servers critical bottlenecks as the number of Internet users increases considerably. Also, authorities may apply certain censorship policies to the provided services of centralized systems. In contrast, such an attempt in P2P approaches is significantly more difficult. Further, today's typical users are equipped

with very powerful machines compared to the past. Such end-point devices are even capable of providing demanding services to a certain extent. Exploiting their integration in service provisioning can decrease significantly the service cost. Harnessing the power of such devices has been the focus of a large part of the distributed systems research community [Ora01].

Distributed systems adopting the P2P paradigm pose significantly more complex communication requirements than their C/S counterparts. Many pragmatic difficulties, e.g., the presence of end-point devices with heterogeneous network, storage and computing capabilities make the development of P2P systems more challenging. More importantly, fundamental difficulties are introduced by three factors: (i) asynchronous concurrent events, (ii) limited local knowledge and (iii) arbitrary failures. In addition, multi-party services raise the need for flexible communication patterns among the participating entities. In order to address the aforementioned issues, P2P systems develop dedicated virtual networks, the so-called *overlay networks*, on top of the physical telecommunication networks [LCP⁺04] [CF04]. Overlay networks are a mandatory abstraction from physical networks to both flexibly fulfill functional requirements such as connectivity maintenance, indexing and routing, as well as to satisfy non-functional requirements, such as scalability, fault-tolerance and load-balancing.

The goal of this thesis is to investigate the design of overlay networks and to quantitatively evaluate efficient and adaptable topologies as well as communication mechanisms that effectively address a large number of conflicting requirements for *widely deployed, large scale, dynamic and heterogeneous* P2P systems.

1.2. Motivation

It has become clear from the preceding overview that P2P systems differ in their characteristics from their C/S counterparts. P2P systems offer a great potential for a new era in distributed computing. However, currently deployed systems are mostly limited to file sharing applications. There are many reasons that hinder the deployment of P2P systems in a much wider range of application domains, mainly related to their complex nature and the achieved dependability and performability. It is the aim of this thesis to address these issues and provide solutions for a new generation of P2P systems. However, P2P systems are not the suggested solution for every distributed system. Instead, a careful study can shed some light in whether it is preferable to follow the C/S or the P2P paradigm [RBDR⁺04].

A vital abstraction of P2P systems is the employed overlay network topology, which is supplied by embedded protocols, mechanisms and algorithms. Several approaches have been proposed as communication schemes in order to provide efficient and scalable inter-peer communication. These schemes are designed on top of the physical networking infrastructure as overlay networks. Their topologies and operating mechanisms influence greatly the performance of routing and topology maintenance algorithms and hence, the efficiency of the involved P2P system.

However, designing such overlay networks is a challenging task. There is a set of requirements that has to be met in designing overlay networks, which may be effortlessly deployed on top of the Internet while making use of available resources optimally. Present overlay design approaches mostly concentrate on a subset of these requirements and ignore the importance of others. Such single-targeted approaches are usually inflexible in adapting to emerging requirements in a later step without compromising the already fulfilled ones. Complete redesign is usually the only solution.

This argument may be demonstrated by briefly analyzing some well-known proposals. Gnutella [Gnu05a] for instance offers a non-hierarchical P2P approach with minimum maintenance cost. However, the employed *flooding* mechanism used for querying makes Gnutella unscalable [Rit01] and caused a system breakdown in the end of 2000 when the number of users increased considerably. Learning from the Gnutella approach two main directions have been followed by P2P overlay network designers.

The first approach is represented by the redesign of P2P systems using *hierarchical* architectures. However, it is questionable whether the existing hierarchical approaches may be still called P2P. The concept of *super-peers*, i.e., peers with additional capabilities, is introduced in these systems. Super-peers form usually the backbone of the overlay network having normal peers placed around them. The architectures of eDonkey [eDo05] and KaZaA [LRW03] may be considered typical representatives of this design direction. The basic drawback of this approach is the requirement for the existence of super-peers, which subsequently imposes new requirements on the system. Super-peers have to operate legitimately since their actions have greater effect in overlay operations. Further, super-peer failures are more severe than normal peer failures. Also, malicious behavior of super-peers has far greater impact compared to the behavior of peers in non-hierarchical P2P systems. Additionally, there is lack of incentives for super-peers to serve the rest of peers. Super-peers may also become performance bottlenecks if there is not a sufficient number of them.

The second approach is based on *Distributed Hash Tables* (DHTs). As their name implies, they are distributed structures that use hash functions to index items that should be mapped on the overlay networks. Globally Unique Identifiers (GUIDs) are assigned to peers, which take values from the same key space as the indexed advertised resources. A minimum distance function is usually used to map items to responsible peers. DHT-based systems aim at providing very efficient routing of messages to their logical destinations. The cost of the routing mechanism grows logarithmically with respect to the size of the overlay network for most of the proposed approaches. This worst case bound applies usually both to the number of hops required for routing in the overlay network as well as the size of the routing tables at peers. Thus, each node is assumed to have a relatively big number of connections (for large-size systems) that grows logarithmically with the system size. Although this requirement increases the robustness of the overlay network, it may be hard to be fulfilled by nodes with low physical capabilities. Further, this solution may become very inefficient in scenarios where high peer join/leave rates occur (which have been empirically observed). Typical representatives of this approach are Chord [SMLN⁺03], CAN [RFH⁺01] and Pastry [RD01a].

In order to overcome the large maintenance cost caused by the high attrition peer rate

in the overlay network and the logarithmically increased connectivity size (with respect to the network size) some alternative approaches were proposed only recently. Koorde [KK03] (a variation of Chord) employs de Bruijn graphs [dB46]. The advantage of this approach is the constant number of connections per peer. Though, this comes at the cost of decreased robustness and a difficulty in incrementally extending the network size. A logarithmic connectivity-based version of Koorde overcomes the robustness problem and provides optimal lookup operations. However, it comes at the cost of high maintenance overhead. A different approach, named Viceroy [MNR02], is based on the butterfly graph. Viceroy requires only a constant number of neighbors with high probability. Though, the construction and maintenance procedures are relatively complex. This brief survey of available overlay networks for P2P systems reveals their shortcomings and identifies the unaddressed challenges found in their design.

1.3. Problem Statement and Goal

As it has been discussed in the Preface and Section 1.1 of this dissertation, the P2P paradigm offers characteristics not present in the C/S paradigm. Moreover, the aforementioned brief survey of available overlay networks for P2P systems reveals their shortcomings and identifies the unaddressed challenges found in their design, thereby, motivating the goals of this thesis. Existing solutions fail to meet in their entirety essential requirements such as scalability, evenly distributed workload and support for heterogeneity, thus, mandating for a carefully investigated approach with high degree of adaptability in dealing with their intrinsic tradeoffs. *The problem tackled by this thesis is addressing the shortcomings and the unsolved challenges identified in the existing structured and hybrid P2P overlay networks.*

Accordingly, the goal of this thesis is to investigate and devise novel *design mechanisms* for P2P overlay networks as well as to develop an efficient and adaptable overlay network *architecture*. Moreover, it aims to develop the relevant *communication mechanisms*, which effectively address a large number of conflicting requirements for *widely deployed, large scale, dynamic and heterogeneous* P2P systems. The proposed architecture and mechanisms are quantitatively evaluated and compared with reference to alternative existing solutions in several scenarios.

1.4. Approach

In this thesis we investigate mechanisms for overlay networks, which follow a hybrid approach in their design. The word *hybrid* is used in many disciplines, e.g., in biology, in sociology, in linguistics, etc. In general, it is used to characterize "*something derived from heterogeneous sources or composed of incongruous elements*" (Oxford Dictionary). Though initially in the context of P2P systems the term *hybrid P2P system* was used to describe approaches that combined both P2P and C/S aspects, its usage was broadened to cover further combinations of heterogeneous approaches. The hybrid approach selected for our

system combines the benefits of DHTs with role specialization methods into well-defined groups of peers.

In general, carefully designed hybrid systems are claimed to be intrinsically better than pure approaches. They allow for the synergistic combination of two or more techniques with additional strengths and reduced weaknesses than either technique alone. Nevertheless, not every arbitrary system combination provides superior solutions but careful selection is required.

In order to justify our decision to follow a hybrid approach, we compare them with non-hybrid solutions in a general, abstract way, avoiding references to specific concrete systems. On the one hand, hybrid systems have increased complexity since they are combinations of more than one approaches. Moreover, they are merged in a possibly constrained way that reveals the advantages of each sub-component. Hybrid systems naturally follow in time the non-hybrid approaches that should be first well understood and both their benefits and drawbacks have to be established. On the other hand, hybrid systems may be better designed since their designers learn from the limitations and the mistakes of the pioneered pure approaches. Carefully designed hybrid solutions can show high adaptability to environmental conditions. Usually, they are designed considering these conditions and they may reveal certain scenario-aware advantages or avoid related limitations. Performance may be greatly increased and new characteristics may be added to the constrained pure alternatives. Apparently, as it will become clear throughout this thesis, hybrid approaches are the only viable way to address the large number of (usually) conflicting requirements raised in large-scale, dynamic and heterogeneous P2P systems.

More specifically, the rationale in our approach is to reduce the high maintenance cost by having a small, fixed number of inter-peer connections and subsequently routing table sizes (at least for the majority of the peers). For this reason the usage of appropriate dense graph structures is suggested. Additionally, clustering mechanisms are utilized to provide a stable network topology. Going a step further a role-based scheme is introduced to deal with the heterogeneity of peers' capabilities and user behavior. This scheme fits the contribution of each node to its capabilities and aims at the maximization of the cluster efficiency by providing appropriate incentives for each role. Thus, it effectively balances between supporting heterogeneity and evenly distributing the load balance.

The predictability of peers' behavior is a challenging issue. Improper assignment of demanding roles to unreliable peers may result in a highly ineffective system performing worse than equally assigning every role to each peer. In order to address this issue we use the *burn-in* approach. Burn-in is a standard engineering method used to identify defected components used in integrated circuits (ICs) production. Using burn-in, engineers can test the newly manufactured components under certain environmental conditions and remove the defected components. Similarly, we capitalize on the observed empirical measurements, which indicate that the probability of peers departing from the system (departure or failure rate) is very high at the initial phase after joining the system and decreases as the time proceeds. We invest on *conditional reliability* techniques to stochastically both increase the endurance of the formatted clusters and minimize the maintenance cost.

Further, advanced mechanisms are applied to reduce the operation cost, by exploiting the

details of their communication scheme. For instance, an efficient *caching* mechanism is developed to reduce the routing effort in scenarios where queries are unevenly distributed. The developed approach does not require additional signaling protocols and may be incrementally deployed to any structured P2P overlay network since it is activated by merely modifying locally the routing semantics. The mechanism considers in its design the observed peer reliability measurements to avoid providing inconsistent cached information.

1.5. Contribution

As described in the previous sections, this dissertation addresses the challenges in designing overlay networks for large scale, dynamic and heterogeneous P2P systems. Our contributions may be conceptively categorized and summarized as supplied in the following list.

1. **Requirements and design space.**

The first step in designing an efficient system is to analyze the underlying requirements. By investigating a set of relevant applications, a rich set of mandatory features has been identified, which is needed for the valid operation of these applications in a large-scale, dynamic and heterogeneous environment.

At a step before designing a novel architecture, we evaluate a plethora of existing solutions against meeting the complete requirements set. In order to analyze the different approaches systematically, a three dimensional design space is introduced to classify the studied approaches.

2. **Architecture.**

A novel, hybrid topology architecture for large-scale, dynamic and heterogeneous P2P systems is introduced. The well designed features of the architecture provide the means to meet a large set of non-functional requirements. In addition, these features effectively balance the potential trade-offs and allow to introduce additional requirements, currently not considered. Scalability, expandability, fault-tolerance, efficiency, load-balance and support for heterogeneity are the major requirements tackled by this architecture. Moreover, the adaptive balancing of their potential trade-offs is particularly considered.

3. **Role model, peer behavior and optimal role assignment.**

Addressing peer heterogeneity in a fair and well-balanced way is a challenging problem. We suggest a novel **rich role model** that maximizes the matching of peers' physical capabilities and user behavior to the requirements of the roles. Peer roles are identified based on the core functionality needed in each P2P overlay network by decoupling the provided operations, i.e., message routing, item caching, advertisement indexing and topology maintenance.

An exhaustive investigation of the available empirical observations on peer availability has been performed aiming to characterize statistically the distribution with para-

metric formulae. The lack of unique and widely acceptable parametric description in the relevant literature motivated the development of a **peer behavior model**. Using this model we explain the intrinsic characteristics of the underlying process that drives peers' uptime by taking into account practical aspects such as the heterogeneity of the peers. We have suggested a mixture of lognormal distributions to accurately describe it.

Further, we have investigated in depth the details of applying the burn-in method in the construction of efficient and reliable P2P overlay networks. A critical issue that arises is selecting the **optimal time to assign a demanding role** to a peer, so that both the risk of failure will be significantly decreased and the system will benefit maximally from the reliability of the stable peers. A condition to effectively apply the burn-in method is to have a decreasing failure rate in the peer lifetime. A thorough exploration of the related reports in the literature acknowledges this fact. A cost model has been constructed to provide the optimal burn-in time.

4. Caching mechanism.

In addition, a simple though efficient caching mechanism is suggested, which capitalizes on the adequateness of caching resources following non-uniform distributions. Additionally, the approach exploits the considerable interest of the P2P users in a relatively small subset of the available resources. The proposed solution extends the capabilities of structured overlay networks without any additional maintenance effort and very low additional routing cost compared to the original algorithms of structured networks. The solution has been explicitly designed in a way where no extension of the signaling protocols is required, thus, avoiding further increase of the operation complexity found in structured overlay networks.

5. Simulation framework and performance evaluation.

Apparently, simulation is the most reliable and useful approach to evaluate complex scenarios for large-scale P2P overlay networks. We have developed a general-purpose open architecture simulator for P2P overlay networks to quantitatively evaluate several network architectures, protocols and mechanisms. The tool utilizes the aforementioned rich role model and separates the provided functionality in three layers (user, overlay network and underlay network). Chord [SMLN⁺03], Gnutella [PSAS01], JXTA [TAA⁺03] and Omicron [DMS04] have been effectively ported into our simulator.

6. Performance evaluation.

The developed P2P overlay network is quantitatively assessed with multiple simulation experiments shedding light on its performance. The most essential mechanisms are evaluated and compared with well-known alternative solutions, i.e., Chord. The characteristics of specific graph structures are quantitatively evaluated and compared with theoretical approximations. Moreover, multiple sampling algorithms are assessed on their ability to cover evenly the utilized graph structures. The observed distributions are approximated with parametric formulae in order to reveal further significant properties.

1.6. Outline

The content of the dissertation is divided in nine chapters. Following this introductory chapter, Chapter 2 supplies the functional and non-functional requirements, which are identified in the context of P2P applications representing interesting use cases. The identified requirements form the set of metrics for evaluating the adequateness of existing P2P overlay networks. Existing solutions are classified based on their design mechanisms and are tested against the requirements to reveal their advantageous features, as well as their shortcomings and limitations in Chapter 3. Special attention is paid in hybrid approaches that appear to offer richer adaptability.

Thereby, a novel hybrid architecture for large-scale, dynamic and heterogeneous P2P overlay networks is developed in Chapter 4. This chapter offers the big picture of the system and introduces all the relevant mechanisms, which are investigated in detail in the subsequent chapters. The details of the identified core roles are given in Chapter 5. There, the utilized algorithms, protocols and mechanisms are described. In addition, common inter-role scenarios are discussed in order to comprehensively explain their relationship. Afterwards, Chapter 6 investigates a user behavior model to describe adequately the peers' reliability and a cost-effective optimization method for efficient assignment of demanding roles.

Thereafter, the open architecture P2P simulator is discussed in Chapter 7. The software architecture of the tool, the provided functionality, the way users can extend the simulator and add new overlay networks as well as the achieved performance are the focus of this chapter. Then, the results collected by using the simulator are provided in Chapter 8. Several experiments have been performed to accurately quantify the performance of our system against a well-known existing solution (i.e., Chord). We conclude our work and present directions of potential future research in Chapter 9.

Overlay Network Requirements

It concerns us to know the purpose we seek in life, for then, like archers aiming at a definitive mark, we shall be more likely to attain what we want.

ARISTOTLE

Conceptual outline.

In order to design a meaningful and useful overlay network, it is imperative to investigate the requirements of P2P systems. In particular, it is necessary to consider the requirements of several important applications and services to identify the common core functionality that has to be provided. File sharing services are without doubt the most popular services deployed over P2P systems; thus, they are considered first in order to identify the critical requirements. Further, a promising as well as challenging application type that is becoming more and more significant is massive multi-player games deployed over the Internet. Such applications have similar requirements as large-scale Collaborative Virtual Environments. We further examine the requirements of such applications and select scenarios and use cases where it is meaningful to opt for a decentralized P2P based approach over a centralized C/S one. The focus is both on functional requirements (common core operations and services that should be provided to the applications) as well as non-functional requirements. Apparently, it is the latter type of requirements that raise challenging issues in meeting them and addressing their (frequently) conflicting nature.

This chapter is organized as follows. Section 2.1 motivates the necessity of identifying the system requirements before we attempt to provide any solution. Afterwards, Section 2.2 describes two interesting application types and identifies the required functionality. Then, important concepts from graph theory that find wide applicability in the design of communication networks are supplied in Section 2.3. Having provided the necessary background, we investigate the functional and non-functional requirements of P2P overlay networks in Section 2.4 and Section 2.5, respectively. The probability of facing constraints in meeting the complete set of requirements and the need for effective trade-offs are discussed in Section 2.6. The chapter is summarized in Section 2.7.

2.1. Motivation

Considerable research effort has recently been devoted to the design of structured peer-to-peer overlay networks resulting in a plethora of approaches. However, not all of them offer innovative features. In fact, a large subset of them focuses merely on scalability, ignoring to address further important requirements raised by large-scale, dynamic and heterogeneous environments. Thus, in order to avoid designing a system not addressing the relevant issues, this chapter aims at presenting the most important requirements that should be considered in the design of overlay networks. It is important to consider real scenarios in the analysis to ensure the usefulness of our approach. Therefore, we consider two significant application types (file sharing and Collaborative Virtual Environments) to identify the crucial requirements. Also, we study their potential cross-effects that are caused by certain combinations of solution mechanisms. Some initial work in supplying a benchmark-based approach for evaluating structured overlay networks is provided in [RRK03]. However, further steps are needed to supply a useful application-driven benchmark-based tool.

The main task in designing an overlay network is constructing the graph that represents the topology characteristics (the nodes and their connectivity pattern). In order to provide an efficient solution the designers have to consider the core (basic) services and operations that may be deployed over this graph. Core services and operations are for instance the routing of messages or the maintenance of the desired graph structure. As an example of the close relationship between the graph structure and the operation of core services consider the (trivial) case of a graph that forms a star. In this topology, the central node represents a so-called communication hot-spot and a central point of failure. For this reason, both static and dynamic characteristics of the graphs have to be considered. Further, the additional requirements that are imposed by the deployed core services of the overlay networks are examined. Service deployment and related requirements have been already addressed in telecommunication services [ITU94], therefore, we consider the collected knowledge in order to define the requirements for this work.

2.2. Applications and Services

2.2.1. File Sharing Applications and Services

In the pre-Napster era, file sharing was mostly performed using centralized approaches such as HTTP and FTP services or decentralized ones such as newsgroups. However, the revolution accomplished by the wide utilization initially of the Napster¹ system and later of other P2P systems following more comprehensively the P2P paradigm resulted in popular file sharing systems, which are generating the majority of the traffic in today's Internet. As an example, some network traffic reports from the Abilene backbone network are shown in Figure 2.1(a) [Abi04]. Also, the traffic measurements shown in Figure 2.1(b) [Has04] demonstrate the importance of file sharing application.

¹The architecture of Napster is not a purely P2P system but a hybrid combination of both P2P and C/S paradigms. Chapter 3 discusses multiple effective combinations of hybrid solutions.

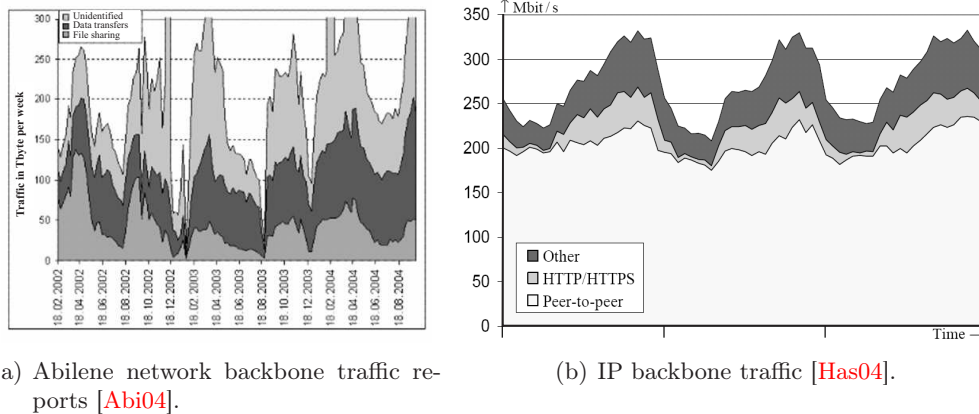


Figure 2.1.: Backbone traffic measurements.

We have analyzed the requirements for a file sharing scenario [DLM03] within the MMAPPs project [Str04]. The following list summarizes the general functionality needed by every P2P file sharing application.

- **System participation.** Users must be able to join the system initially. A bootstrap phase may provide an active peer to forward the join request to. Since peers are autonomous entities departure can simply happen by disconnecting from the system (though this may have considerable consequences to the state of the system).
- **Sharing files.** Users must be able to select which files they wish to advertise so that remote users may find them.
- **Providing files.** When a remote user requests a local file, this file must be uploaded. Scheduling mechanisms may provide fair sharing of local resources when the demand is high.
- **Finding files.** Finding files is a required step before a user may be able to download it. Assuming a large-scale system, global directories may not be a viable option in providing this function.
- **Rating files.** Since users may arbitrarily share files of unknown content and quality, a rating system may be needed to collect information about the shared files and the relevant users.
- **Parallel downloading.** Efficient download performance is very crucial for this application. Downloading in parallel from multiple users may increase the achieved throughput.
- **Monitoring contribution.** Monitoring the contribution of each peer is an essential feature, especially in cases where users are reluctant in providing local resources voluntarily. Advanced accounting systems may be used to provide fairness and efficiency [LDMS05].
- **Forming subgroups.** The ability to form user subgroups may allow the creation of special communities with rules accepted by all members.

However, not every single function of the above list must necessarily be offered by the overlay network. Certain operations are end-to-end issues, involving only the service provider(s) and the service consumer(s). Also, some functions are offered by advanced services that may be designed on top of the basic primitives provided by the middleware. Further, some aspects like parallel download may become more efficient if the overlay structure is involved, however, such functionality is out of scope of this thesis. Content replication [On04] may increase the availability of specific content and provide better quality services. In general, Content Management Systems (CMS) [MT04] is a crucial and interesting application area that may benefit by P2P technologies.

In addition to the above functionality, there is a set of non-functional requirements that must be fulfilled by the P2P system. This set includes scalability, fault-tolerance, load-balance, support for heterogeneity and efficiency. These requirements will be discussed in later sections of this chapter.

2.2.2. Beyond File Sharing

Seeking to examine an interesting application type beyond file sharing, we selected to examine the adequateness of P2P technology in Collaborative Virtual Environments and Massive Multiplayer Games.

Collaborative Virtual Environments (CVEs) are very demanding systems, since they integrate multiple media types with various demanding network requirements and a large number of frequently interacting users. Several architectures have been proposed in the past to deal with the communication overhead and the management of the collaboration sessions. Basically, two major directions have been followed: (i) the traditional C/S paradigm and (ii) the all-to-all server-less approach. However, both of them have limitations in terms of scalability. C/S approaches require additional server resource capabilities, as the number of clients increases. All-to-all approaches have high bandwidth requirements in order to enable communication among the participants and even harder to solve problems, such as global information consistency. Further, additional issues have to be addressed, e.g., fault-tolerance or heterogeneity of the participants. Network-layer multicasting could (if available) improve the efficiency of the all-to-all approach.

Additionally, a significant research effort has been focusing lately in designing *Massive Multiplayer Games* (MMGs) based on P2P technology. The game industry seems to be an important driving factor for the future research on this field. In particular, Knutsson et al. [KLXH04] developed SimMud, an experimental MMG prototype on top of FreePastry (an open source P2P platform based on Pastry [RD01a]). FreeMMG [CBG03] is an open source initiative to support MMGs. Also, a scalable publish/subscribe system for Internet games called Mercury [BRS02] has been developed, and Iimura et al. [IHK04] developed a system to support multiplayer online games.

Table 2.1.: CVE content classification.

<i>Synthetic Content Type</i>	<i>Information Type</i>		<i>Driving Requirements</i>	<i>Communication Pattern</i>	<i>Update</i>
• Environment Description	Static		Storage Space	Locally Accessed	Pull
• Events	Dynamic	Transient	Latency, Ordering	Streaming	Push
• State		Persistent	Maintenance, Consistency	Unicast Fetching	Pull

Content description and characteristics

The media sources that compose the rendered scenes in CVEs have heterogeneous characteristics and delivery requirements. Here, we focus merely on the maintenance and delivery of *Synthetic Graphics Media Sources*. Table 2.1 provides a coarse classification of the synthetic graphics content, expanded to include the related characteristics and delivery mechanisms. A brief tutorial on the essential concepts found in CVEs is supplied in Appendix C in order to provide a smoother introduction to the rest of this section.

- *Environment Description*. The synthetic media content describing CVEs is statically defined. Such content includes immutable landscape description. Landscapes can be defined by vendors that provide the applications or can be user-defined, created by usually available editors. Role Playing Games (RPGs) are common examples of such applications (though the technology used is usually proprietary).
- *Events*. Another important information type (related to the synthetic media content) is the captured interaction of the users with the content itself and their navigation inside the world. Depending on the application, delivery of such events may become very demanding in terms of tolerated latency. However, concepts such as AoI and Locomotion reduce significantly the generated network overhead, making their implementation more feasible. Multicast-based communication mechanisms are common in delivering such content. Users in the vicinity of the event sources subscribe to receive the transient information.
- *State*. An additional dynamic information type is the announcement of the active event sources so that users can subscribe to receive the event streams and the state of (possibly inactive) mutable objects. Mutable object descriptions are initially included and retrieved, similarly to the static content introduced above. However, users can interact with the mutable objects and modify their state. While the interaction itself is described by a series of events available to the subscribers of that source, the result is important to be captured and be available to users beyond the spatiotemporal limits of the Locomotion and time, even when no active user is interacting with that object. Such a spatiotemporal evolution from transient to persistent state information is described in Figure 2.2. As an interesting example, consider a snowed landscape where a couple of avatars is walking on. The two users can immediately observe their footprints left on the virgin snow via the streaming of the generated

events. Assume that shortly after those two avatars left the specific Locale a new one enters that area. The marked footprints on the snow must be available for observation.

Here, we focus on maintaining and providing the last type of the aforementioned information. It should be noted that such type of information triggers the subscription of a listener to transient information generated by the media source, if the source is active. In addition, *event streaming* may be an interesting use case to examine, however, it is out of the scope of this thesis. Multicast mechanisms either native [Rim04] or overlay based may be required. Physical network support by ISPs [Hec04] can improve considerably the performance.

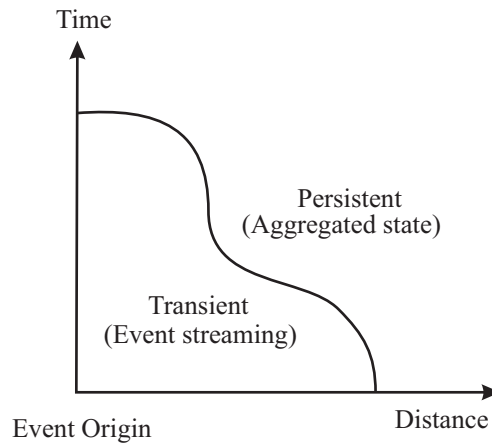


Figure 2.2.: Evolution of event handling.

Usage Patterns

A very important factor in designing an optimal architecture for CVEs is to take into account the way information is required at and updated by the participating users. As it has been explained in Appendix C, Locales partition the virtual world in order to reduce the required information that has to be delivered to each end-point device. However, as users move from Locale to Locale, they should be informed about the status of all mutable objects that belong to the entering Locale. Frequently changing Locales increase significantly the information that should be delivered, thus, introducing a potential bottleneck for centralized approaches. In such scenarios, P2P approaches may perform more efficiently assuming an appropriate overlay network design. In contrast, mutable object states are updated at rates that satisfy the needs of the application, whenever users interact with them. *Pre-fetching* techniques may be used to reduce the probability that the lookup query in the P2P network may be delayed more than the tolerable time threshold. Summarizing, it is common that while update operations are performed on single objects, read operations are grouped to all the objects of a Locale.

2.3. Graph Theory and Networks

Graph theory concepts are widely used to evaluate the static (as well as the dynamic in many cases) properties of communication networks. It is necessary to supply the crucial graph theory term definitions, before studying the relevant requirements of P2P overlay networks. Further knowledge on the topic may be gathered from graph theory textbooks such as [Jun99], [Ber62], [Kau67], [Min78].

A *graph* G is a pair $G = (V, E)$ consisting of a set $V \neq \emptyset$ and a set E of two element subsets of V . The elements of V are usually called *vertices* in graph theory; however, throughout this dissertation, the terms *nodes* or *peers* are used instead. An element $e = \{u, v\}$ of E is called *graph edge* with *end vertices* u and v . We say that u and v are *incident* (alternatively *adjacent* or *neighbors*) to each other. The terms *connection* or *link* are also used to describe an edge in this dissertation. Edges can be *directed* (uni-directional) or *undirected* (bidirectional). A graph in which each graph edge is replaced by a directed graph edge is also called a *digraph*.

For any vertex u of a graph, the *degree* ($\deg u$) of u is the number of edges attached to u . A graph is said to be *regular* of degree r if all local degrees are of the same number r . The *order* or the *size* $|V|$ of a graph V is the population of its vertices. A *path* in a graph is defined as a sequence of nodes where each element is adjacent to the previous (with the exception of the first). The *distance* $d(u, v)$ between two vertices u and v of a finite graph is the minimum length of the paths connecting them. If no such path exists (i.e., if the vertices lie in different unconnected components), then the distance is set equal to ∞ . The length $\max_{u,v} d(u, v)$ of the "longest shortest path" between any two graph vertices (u, v) of a graph is called the *diameter* of the graph. A *cycle* of a graph G , sometimes also called a *circuit*, is a subset of the edge set of G that forms a path such that the first node of the path corresponds to the last. *Girth* is the length of the shortest graph cycle (if any) in a graph. An interesting metric to evaluate the resilience of a graph is the *bisection width* [Lei91], which is defined as the minimum number of edges between any two equal-size partitions of the graph. The bisection width metric can determine the resilience of a graph to be split into large components by removing individual nodes.

Digraphs have been extensively used in interconnection networks for parallel and distributed systems design (cf. [HW97], [LS96]). Digraphs received special attention from the research community aiming to solve the problem of the so-called (k, D) digraph problem. The goal is to maximize the number of vertices N in a digraph of maximum out-degree k and diameter D [FYdM84]. Some general bounds relating the order, the degree and the diameter of a graph are provided by the well-known Moore bound [BT80]. Assume a graph with node degree k and diameter D ; then the maximum number of nodes (*graph order*) that may populate this graph is given by Equation 2.1:

$$N \leq 1 + k + k^2 + \dots + k^D = \frac{k^{D+1} - 1}{k - 1}. \quad (2.1)$$

Interestingly, the Moore bound is not achievable for any non-trivial graph [BT80]. Nevertheless, in the context of P2P networks, it is more useful to reformulate Equation 2.1 in a way that provides a lower bound for the graph *diameter* (D_M), given the node degree

and the graph order [FL92]:

$$D_M = \lceil \log_k(N(k-1) + 1) \rceil - 1 \leq D. \quad (2.2)$$

The *average distance* (μ_D) among the nodes of a graph may also be bounded by the following inequality [SR94] (which is approximated in [LKR03]):

$$D_M - \frac{k(k^{D_M} - 1)}{N(k-1)^2} + \frac{D_M}{N(k-1)} \approx D_M - \frac{1}{k-1} \leq \mu_D. \quad (2.3)$$

Further, a d -regular graph is a (d, c) -expander or has a c -expansion (for some positive c) if and only if for every subset $U \subset V$ of size at most $|V|/2$. Expander graphs have been investigated also for P2P overlay networks [GMS04], [LS03], [GPU01], particularly focusing on their ability to expand exponentially by increasing linearly their diameter.

Two particular classes of graphs have been adapted by several researchers for overlay network design: The well-known *small-world* graphs and the *scale-free* graphs. Their properties fit well with the effective operation of loosely structured overlay networks. An excellent survey on the issues of aforementioned graphs can be found in [WC03]. However, lexicographic (i.e., de Bruijn) graphs [dB46] have been found to have more promising properties and they employed as the core network structure of the proposed architecture.

2.3.1. Small-world Networks

Often in the discussion between unfamiliar people the expression "*What a small world!*" is used to describe the discovery that they have a common friend or relative. Apparently, social experiments have been performed examining the human relationships and the degree to which the *small-world phenomenon* can be observed [Mil67]. Moreover, similar associations have been observed in a number of complex systems, e.g., the World Wide Web [Ada99] and the human neuroanatomy system [SCKH04]. Several researchers have investigated the properties and the underlying organizing principles of these networks [Kle02], [Kle99], [WC03], [New00].

From a graph theoretic point of view, a small world network is a graph exhibiting a high clustering coefficient, but low characteristic path length (diameter). More precisely, the *clustering coefficient* CC of a graph is the metric to characterize a network as having the small-world property. It is defined as the average fraction of pairs of neighbors of a node that are also neighbors of each other. Suppose that a node i in the network has c_i neighbor nodes. Apparently, at most $c_i(c_i - 1)/2$ edges can exist among them, and this occurs when every neighbor of node i is connected to every other neighbor of node i . The clustering coefficient CC_i of node i is then defined as the ratio between the number E_i of edges that actually exist among these c_i nodes and the total possible number:

$$CC_i = \frac{2E_i}{(c_i(c_i - 1))}. \quad (2.4)$$

The clustering coefficient CC of the whole network is the average of CC_i s. Obviously, $CC \leq 1$ where the equality holds if and only if every node in the network connects to

every other node. In a completely random network consisting of N nodes, $CC \sim 1/N$ (which is very small as compared to most real networks). On the other hand certain structures of regular graphs have a high clustering coefficient. Typically, small world networks combine two characteristics that are attractive for P2P systems (i) relatively low diameter and (ii) high connectivity (resulting from the high clustering coefficient).

The properties of small-world networks have been exploited in the context of P2P networks by multiple approaches, e.g., the DIET network [HWBM02], the Swan network [Bon02], the Symphony network [MBR03], as well as in an improvement of the original Freenet network [ZGG02].

2.3.2. Scale-free Networks

The distribution of the node degree in a network is a significant factor for several network properties. A number of real life complex networks, e.g., income of individuals, genera, Internet file sizes [RH02], the World Wide Web, metabolic systems, paper co-authorship, movie actors [AB02], cognitive sciences [SCKH04], etc. appear to have a power law² distribution of the form $P(k) = k^{-y}$ where for the most typical cases $1 \leq y \leq 3$ [WC03].

Apparently, for one of the most well-known P2P networks, the Gnutella network, it has been shown that its nodes follow a power law distribution (though not built implicitly into its design) [ALPH01], [Kab01] [PSAS01], [NG01], [ALH02]. The underlying mechanism for this evolution is the *preferential attachment* mechanism, where the probability Π_i of connecting to peer i with degree c_i is $\Pi_i = c_i / \sum_j c_j$. Peers tend to connect to well-known peers with higher probability ending up in the so-called "*rich get richer*" phenomenon.

Networks that have power law distributed node degree are called *scale-free* networks. While such networks have desirable characteristics such as relatively small diameter and can effectively support heterogeneity, they are vulnerable to attacks and diseases dissemination [Ald03], [BB03]. Moreover, a node with high degree holds an important position in the network. A possible removal of the node can drastically change properties such as the diameter and might result in multiple smaller graphs (*network fragmentation*). Moreover, if the graph is considered as a communication network, nodes with high degree are involved in delivering a large amount of traffic, ending up to potential traffic "*bottlenecks*".

2.3.3. Lexicographic Digraphs

An interesting class of digraphs are the lexicographic digraphs, which includes the de Bruijn and Kautz digraphs [BSGL96]³. de Bruijn graphs belong to a very important class of graphs; they have asymptotically optimal graph diameter and average node distance [LKR03]. Thereby, they are employed in the design of our work. Before we present the way de Bruijn graphs are used in the scope of this work, the topology of those graphs and the way routing algorithms operate are described.

²Refer to Appendix A for more details on the power law distribution.

³de Bruijn graphs are less dense than Kautz graphs but they are more flexible since they do not have any limitations on the sequence of the represented symbols in every node.

Figure 2.3 shows a directed de Bruijn(2,3) graph denoting a graph with a maximum out-degree of 2, a diameter of 3 and order 8. For graphs with fixed out-degree of 2 the maximum number of nodes is always limited by 2^D ⁴. The graph contains 2^{D+1} directed edges in this case. Each node is represented by a D -length ($D = 3$ in this example) string. Every character of the string can take k different values (2 in this example). In the general case, each node is represented by a string such as $u_1u_2...u_D$. The connections between the nodes follow a simple left shift operation from node $u_1(u_2...u_D)$ to node $(u_2...u_D)u_x$, where u_x can take one of the possible values of the characters $(0, k - 1)$.

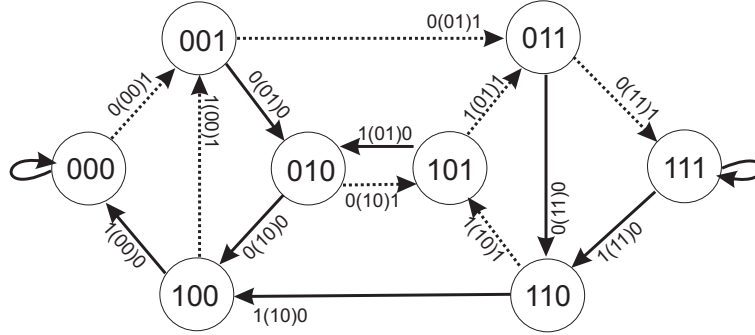


Figure 2.3.: Directed de Bruijn(2,3) graph.

de Bruijn graphs have been suggested to model the topology of several P2P systems, though, as it will become clear in the consequent chapters, we exploited them in a different way. Considerable examples of P2P systems that use de Bruijn graphs are Koorde [KK03], D2B [FG03a] and Optimal Diameter Routing Infrastructure (ODRI) [LKR03].

2.4. Overlay Functionality

The common supplied functionality of P2P overlay networks is mainly focusing on the following functionality discussed in this section. The provided operations may be divided in two categories, based on the criterion whether it is a user-triggered functionality or system-triggered functionality. A *user* may not necessarily be a human being, but instead may be a bot or an agent or an advanced service or application that is designed to use system's functionality. System-triggered functionality consists of activities that aim at achieving efficient system operation. Table 2.2 summarizes the common functionality. More advanced functionality such as application layer multicasting, anycasting or mobility may be offered by higher infrastructure levels such as the Internet Indirect Infrastructure (I3) [SAZ⁺04], which is built on top of Chord.

⁴The Moore bound determines always maximum upper bounds on the size of the graphs that are not reachable for non-trivial cases.

Table 2.2.: Classification of common overlay operations.

Category	Operations
<i>User-triggered</i>	Joining to and leaving from P2P overlay networks Searching for available resources and services Advertising local resources and services Selective information dissemination (publish/subscribe)
<i>System-triggered</i>	Topology maintenance for minimum operational cost Index caching for popular advertisements Notify user about critical network events

2.4.1. User-triggered Operations

As a first step, a peer must join a P2P system. A bootstrap phase⁵ provides the necessary "hooks" to initiate the joining procedure. Common approaches met in widely deployed P2P systems include *history-based* mechanisms, *word-of-mouth*, well-known public Web sites and well-known *seed* peers. Another alternative for particular cases may be the multicast-based mechanism suggested in MPEG-4 DMIF [ISO99b], where users disseminate their existence in well-known multicast channels. Assuming that a peer has successfully obtained another valid address of a peer that is currently member of the system, the second phase of joining may begin, where the peer may be randomly or deterministically positioned at the overlay network.

While the join procedure is completed after a number of message-passing steps, in contrast, peers may depart arbitrarily from the system due to their complete autonomy. It may be advantageous for the system to provide incentives to peers to complete any pending operations and notify their neighbors about their impending departure before it occurs. However, such mechanisms are not typically met in most deployed systems.

The most important operation provided by a P2P overlay network is its ability to search for advertised items despite the large size of the system. However, the semantics of the search operation may differ resulting to various realizations. For example, file sharing systems (e.g., Gnutella) usually offer a text-based search operation where the requestor floods the network with messages that include the requested file name (or keywords) [CAN02]. Then, it may receive back replies including advertisements, which have indexing information that matches the query text. However, such an approach may be very costly in terms of required communication load.

In cases where an advertisement may be uniquely identified (e.g., using a "perfect" hash function), more advanced search mechanisms may be deployed. Such search operations are termed as *lookup* operations [BKK⁺03]. The efficiency of lookup operations is limited though in cases where the requested advertisement may be uniquely identified. This shortcoming motivated researchers to investigate advanced mechanisms to provide richer functionality for lookup operations. Such operations are called *complex queries* [HHH⁺02]

⁵A bootstrap phase is required both in the initial P2P system deployment stage as well as in every peer join attempt.

or *range queries* [AX02]. SHARK [MS03b] is such a system, which capitalizes on grouping the users based on their interests.

Nevertheless, before peers are enabled to search for useful advertisements, local peer resources and services must be *advertised* first. The advertisements may be stored locally or remotely (or both). The advertised information may be as simple as a text field or a *globally unique identifier* (GUID) or even rich metadata structures. The indexing information may hold the complete information or some aggregated summary.

Another useful operation is the ability to selectively disseminate information, the so-called *publish/subscribe* systems, e.g., Scribe [CDKR02]. However, since such an operation is similar to multicasting, it is out of the scope of this thesis and is not considered further. Similarly, users may find useful to be asynchronously informed when important events occur in the system.

2.4.2. System-triggered Operations

System-triggered operations aim at maintaining a useful state in the system so that user-triggered operations may be more efficiently performed. The most important system-triggered operation is the *maintenance* of the targeted overlay network topology in order to utilize structure-related benefits. For example, in Chord the complexity of the lookup operation is upper bounded by the logarithm of the number of alive peers in worst case scenarios, as long as the topology is stabilized to the targeted ring with valid fingers⁶. However, proactive mechanisms (mechanisms, which operate in advance targeting to predict the future state, thus, improving system performance) that are frequently used in maintaining the perfect topology may cause significant amounts of traffic in scenarios where peers join and leave very frequently. This aspect will be thoroughly examined in the context of this thesis.

Moreover, *caching* (though an optional operation) may help reducing the operational cost of the system. Apparently, caching is suggested in several distributed system technologies (e.g., Web technologies [AFJ00], or distributed file systems such as AFS [How04]). Appropriate caching policies that consider the intrinsic characteristics of P2P systems have been proven very useful, especially in scenarios where query distributions and resource popularity do not follow uniform distributions. A promising low-cost overlay network-unaware mechanism is discussed in Chapter 5.

For many cases, it is useful to **notify** the user level to get a decision on a critical event, which are of high importance for the user. For example, the user may be informed when particular replies to posed queries are received in order to select further operations.

⁶Chord is described in deeper detail in the related work provided in Chapter 3

2.5. Non-functional Requirements

2.5.1. Scalability

The factors of scale have been investigated in depth in the context of traditional distributed systems. Three basic dimensions have been identified: (i) the numerical dimension, which consists of the number of users, objects and services encompassed, (ii) the geographical dimension, which consists of the distance over which the system is scattered and (iii) the administrative dimension, which consists of the number of organizations that exert control over parts of the system [Neu94]. Modern P2P systems are challenging systems that may scale enormously over all of the three identified dimensions. Peers may be distributed globally and each user may have the absolute control of each own machine. Typical sizes of P2P systems may reach several millions of users and it is envisaged to be extended to billions. A simple definition for *scalability* is the following:

Definition 2.1. *Scalability is the ease with which a system or component can be modified to fit the problem sphere.*

To compare the scalability of various algorithms, a variety of metrics have been developed in the context of massively parallel computation. These metrics assume that the program runs on a set of k processors with a given architecture, and that the completion time T measures the performance. Three related kinds of metrics have been reported: *speedup metrics*, *efficiency metrics*, and *scalability metrics*. The following definitions give the flavor of the proposed metrics [SN93]:

- *Speedup* S measures how the rate of doing work increases with the number of processors k , compared to one processor, and has an "ideal" linear speedup value of $S(k) = k$.
- *Efficiency* E measures the work rate per processor (that is, $E(k) = S(k)/k$), and has an "ideal" value of unity.
- *Scalability* $\psi(k_1, k_2)$ from one scale k_1 to another scale k_2 is the ratio of the efficiency figures for the two cases, $\psi(k_1, k_2) = E(k_2)/E(k_1)$. It also has an ideal value of unity.

However, the significant peer heterogeneity and the highly unpredictable peer failure rate observed in P2P systems make hard to accurately evaluate their scalability with such formal metrics. *Productivity* is a cost-effective scalability metric, which is presented in [JW00]. The system is scalable if productivity is maintained as the scale changes. Given the following quantities: (i) $\lambda(k)$ = throughput, (ii) $f(k)$ = average value of each response/sec, calculated from its quality of service and (iii) $C(k)$ = cost expressed as a running cost/sec to be uniform with λ at scale k , then the productivity $F(k)$ is the value delivered per second and is given by $F(k) = \lambda(k) \cdot f(k)/C(k)$. The scalability metric at two different scale factors k_1 and k_2 is defined as the ratio of their productivity:

$$\psi(k_1, k_2) = \frac{F(k_1)}{F(k_2)}. \quad (2.5)$$

Scalability in the design of overlay networks may be measured both in time and space

dimensions. However, there is always a trade-off. Time complexity involves the number of hops that are required to forward the messages from the source node to the destination. Space complexity is related to the size of the state that each node is required to maintain in order to keep the overlay in a functioning state. This information includes the routing table entries of the neighbor nodes as well as the indexing structures for the advertised items. Additionally, the size of the messages and the overhead of the maintenance procedure should be considered in order to evaluate the scalability of the overlay network.

2.5.2. Expandability

Expandability expresses how a system or a component can be modified to increase its storage, communication or functional capacity at low cost.

Definition 2.2. *Expandability of the topology is the ease with which the topology graph can be expanded to larger sizes.*

The topology is *incrementally expandable* if the definition of the topology allows graphs of any size. Otherwise (if the size granularity is only greater than one) they are defined as *partially expandable*. Some hierarchically recursive topologies allow graphs of specific discrete sizes (such as powers of two). If a topology is incrementally expandable, a very important question is how the structure of an instance of size n differs from the structure of an instance of size $n + k$ for some integer constant $k \geq 1$.

An $(n + k)$ -vertex instance can be obtained from a n -vertex instance by removing $r(k)$ edges (to get a subgraph of the larger instance) and by adding additional vertices and corresponding edges to this subgraph. If $r(k) = O(k)$, the topology is said *incrementally scalable* [Tvr99]. Very few topologies are incrementally scalable. For example, 2-D meshes are incrementally expandable, but not scalable.

2.5.3. Dependability

Dependability is an "umbrella" system property covering multiple attributes such as reliability, availability, safety and maintainability. A systematic explanation of dependability consists of three parts as it is shown in Figure 2.4 [ALR00]. Dependability may be defined as follows:

Definition 2.3. *Dependability of a computing system is the ability to deliver service that can be justifiably trusted [ALR00].*

Dependability has been extensively explored in the context of dependable routing in mobile and ad hoc networks in [Hol04] and has been the topic of extensive studying by standardization bodies such as ITU [ITU94].

Availability, Reliability and Fault-tolerance

ITU-T Recommendation E.800 defines *availability* and *reliability* as follows:

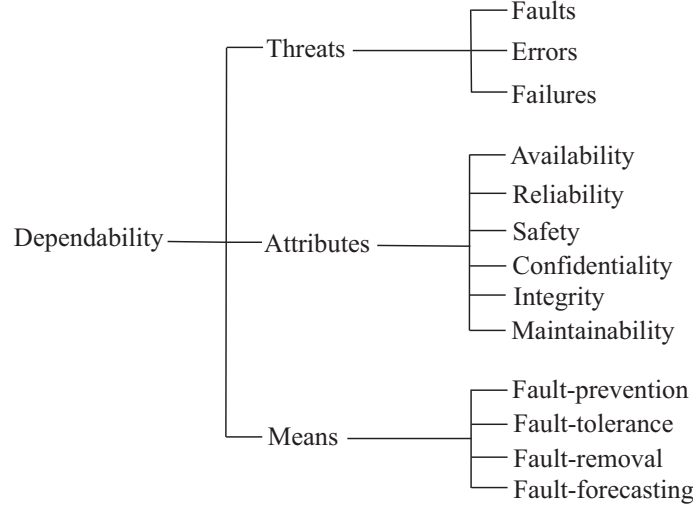


Figure 2.4.: The dependability tree [ALR00].

Definition 2.4. *Availability is the ability of an item to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided [ITU94].*

Definition 2.5. *Reliability is the ability of an item to perform a required function under given conditions for a given time interval [ITU94].*

Reliability $R(t)$ is defined as the probability that the system continues to function throughout the interval $(0, t)$. It is not necessary to (but it is often) assumed that the system is functioning at time 0. Stochastically, let the random variable X be the time to failure of the system. Then, the reliability of the system is defined as:

$$R(t) = \Pr\{X > t\} = 1 - F(t). \quad (2.6)$$

where $F(t)$ is the distribution function of system lifetime. The *Mean Time To system Failure* (MTTF) is defined as:

$$MTTF = E[X] = \int_0^\infty t f(t) dt = \int_0^\infty R(t) dt. \quad (2.7)$$

Reliability is often known as the *cumulative distribution function* (CDF) or the *survival function*. Reliability (though closely related) is distinct from availability. An important difference between the two concepts is that reliability refers to failure-free operation during an interval, while availability refers to failure-free operation at a given instant of time, usually the time when a device or system is first accessed to provide a required function or service. The conceptual difference can be properly visualized with a Markov chain example, as it is illustrated in Figure 2.5. In Figure 2.5(b) state S_0 is an *absorbing* state representing an unrecoverable failure, while S_1 and S_2 are *transient* states. In contrast, state S_0 in Figure 2.5(a) represents a recoverable failure.

In the context of core operations found in structured P2P overlay networks, reliability is a more useful concept expressing more adequately the fact that after peers' departure the

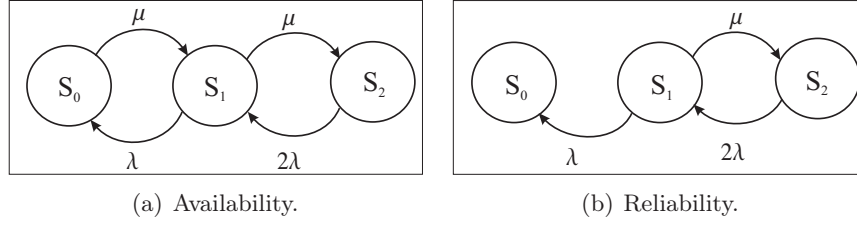


Figure 2.5.: Availability versus Reliability: Markov chain examples.

component is considered as failed and there is no necessity to maintain further its state. Such a modeling approach is adequate for the time scale the shared indexing information is valid. Thus, we will consider the reliability attribute in the following system analysis.

Fault-tolerance is considered as a particular mean to develop the attributes that constitute the dependability concept.

Definition 2.6. *Fault-tolerance is the ability of a system or component to continue normal operation despite the presence of hardware or software failures.*

In terms of overlay networks, it expresses the resilience of the connectivity when failures are encountered by the arbitrary departure of peers⁷.

As it has already been mentioned, P2P systems have been studied extensively [SGG02], [KWX01], [SW04a]. One main aspect among others is the observation of user behavior. An interesting result is the fact that the majority of the peers tend to stay connected with the system for a relatively short time. This results in a highly dynamic system with high join and leave rates, which hinders fault-tolerant routing as it is also discussed in [ADS02] and [HK03]. This kind of behavior imposes another requirement especially important for structured overlays. Effects like overlay partitioning are possible in case where a special overlay maintenance procedure is not in place. High maintenance costs to keep the structure in a *stable* state and provide reliable services are some of the consequences.

Cooperation and Security

Before proceeding with the concept of security in P2P systems, it is crucial to characterize the user behavior in such systems. Relevant peer characterizations are provided in the context of rational behavior in P2P systems from an economy view point [SP03] and in terms of dependable routing [Hol04], [HMKR04], [HSSS04]. Overlay network security is highly relevant to node misbehavior. Revisiting the node classification provided in [Hol04] four different node types may be identified: (i) *Cooperative nodes*, (ii) *Inactive nodes*, (iii) *Selfish nodes* and (iv) *Malicious nodes*. Though the notion of inactive nodes is mostly appropriate for wireless ad hoc networks, it may be of interest to P2P networks both because P2P systems may be deployed in wireless ad hoc networks as overlay networks and because ad hoc nodes have a peer-to-peer relationship at the network level. Nevertheless,

⁷We abstract from the physical network failures

selfish and malicious peers are of prime interest for the dependable and secure operation of P2P systems, so their definition is given. Selfish nodes maximize their own gain.

Definition 2.7. *A selfish node does not forward any data packets for other nodes except for itself. The node cooperates during the route discovery cycle to maintain a correct routing table and to be present in other routing tables [Hol04].*

Malicious nodes reduce the utility of the network without regard for their own gain.

Definition 2.8. *A malicious node abuses the cooperation among nodes to hinder operation of the network [Hol04].*

A useful definition for security is given below.

Definition 2.9. *Security is the ability of a system to manage, protect and distribute sensitive information [MAC04].*

In the context of P2P overlay networks, security issues are basically raised by the presence of malicious peers. Additionally, selfish peers may behave in a way that could have similar results. Castro [CDG⁺02] and Sit [SM02] address the most important security aspects in P2P overlays. They are mainly focusing on the forwarding operation, where malicious peers can either drop the packages or forward them in wrong directions and on indexing responsibilities. Furthermore, [DGM02] focus on the Denial of Service (DoS) attack problem. However, security issues are out of the scope of this thesis.

Persistence, Consistency and Integrity

Advertised content's persistence, consistency and integrity is not a functionality that has to be offered by the overlay network itself. For example, in the case of distributed storage systems, an additional layer is suggested to be used on top of the overlay network, which is responsible to handle such issues. Interesting examples of distributed storage systems are the CFS [DKK⁺01] system, which is built on top of Chord and the PAST [RD01b] system, which is built on top of Pastry. The general principles of distributed file systems are studied in [TN97] and [BDET00].

However, providing consistent and valid indexing information about the advertised resources is a crucial requirement posed on P2P overlay networks. Index replication mechanisms and information updating mechanisms are needed.

2.5.4. Load-balance, Fairness and Heterogeneity

Definition 2.10. *Load-balance in P2P systems is the extent to which the load is evenly spread across nodes.*

The load considered in this context consists of the effort required for the basic overlay operations, e.g., maintenance, routing, indexing, caching, etc. On the one hand, designing an overlay network that avoids hot spots increases the performance and the fault-tolerance of

the overall system. Appropriate mechanisms are required to evenly distribute the common tasks among the peers (e.g., uniform distribution of advertised resources [BSS02]).

On the other hand, by taking into account the heterogeneous environment, not all of the nodes are capable of offering the same amount of resources. A *fair* solution should provide the necessary incentives and the weighted balance between the resource contribution and the consumed overlay services. Fairness has been defined in terms of sharing network resources, e.g., in multicast scenarios [Rim04] or in more general terms in [JCH84].

Overlay network design should take into account the *heterogeneity in the physical capabilities of the peers and the user behavior*. Designing schemes that require homogeneous components can either decrease the system capabilities to those achievable by the weakest components or faulty/inefficient operation should be expected from the least capable nodes. Moreover, the observed variation in user behavior (e.g., up-time patterns) [MTG03] should be taken into account in the design of the overlay to increase the efficiency of the systems.

2.5.5. Efficiency

Efficiency may in general be defined as follows.

Definition 2.11. *Efficiency is the ratio of productive resource consumption and total resource consumption.*

Alternatively, in terms of computer systems the following definition may provide better understanding.

Definition 2.12. *Efficiency is the degree to which a system or component performs its designated functions with minimum consumption of resources (CPU, Memory, I/O, Peripherals, Networks) [Glo04].*

In the context of overlay networks, efficiency may be described in terms of *routing* and *physical network proximity*. The actual packet routing from a source node to a destination node is a service offered by the underlying network [Hec04]. The complexity of this layer (resulting from the IP address assignment procedure) should not be repeated in the overlay network. Combined with an appropriate node identification scheme the overlay routing mechanism should be a simple operation. Based on the destination node address the mechanism may return the closest neighbor to the destination.

*Network proximity*⁸ consideration is a crucial factor in reducing the latency on the network operations. *Relative Delay Penalty* (RDP) is a measure for the additional packet delay introduced by the overlay on the delivery of a message between two nodes. It is defined as the ratio of the latency experienced when sending data using the overlay compared to the latency experienced when sending data directly using the underlying network. *Link stress* is another metric that quantifies the usage of the underlying network from the overlay. It is defined as the number of tunnels that send traffic over a physical link. Links may be "stressed" despite the fact that the overlay traffic is well-balanced among its

⁸ *Locality* and *vicinity* are alternative widely used terms.

nodes. For example, studies of the Gnutella system [SK03], [RFI02] observed that request forwarding performed many "zig-zag" forwarding steps between nodes that are physically located in North America and Europe. Many studies are focusing in optimizing this aspect [RKY⁺02].

2.5.6. Autonomy

P2P systems are composed of a set of independent nodes that are not centrally controlled or administered. Despite this fact many proposals adopted hierarchical solutions for efficiency reasons (e.g., KaZaa [LRW03] and eDonkey [eDo05]). Such systems imposes additional requirements on the supporting infrastructure. In the case of eDonkey particularly the servers run different software. This is an even stronger requirement than potentially assigning a super-peer role to certain capable peers. Flat approaches (either structured like Chord [SMLN⁺03] and Pastry [RD01a], or unstructured like Gnutella [Gnu05a] and Freenet [CSWH00]) are definitely preferable over the hierarchical alternatives since they maintain peer autonomy.

2.5.7. Anonymity

Anonymity in computer systems is defined as follows.

Definition 2.13. *Anonymity is the degree to which a system or component allows for (or supports) anonymous transactions [Glo04].*

This is a special requirement for certain applications that require anti-censorship features or to increase the privacy of the participating users. Still, a node identification scheme is required: Many solutions suggest the modification of certain important header fields during routing in order to make hard the extraction of the identity of the message originator. Of course, misuse of those systems is always an issue. Freenet [CSWH00], Free Haven [DFM00], Achord [HW02], Hordes [SL00], Publius [WRC00], Tangler [WM01] and Tarzan [FM02] are well-known examples of systems with strong anonymity features. Anonymity is not directly addressed by the presented approach and is not in the scope of this thesis.

2.6. Constraint Requirements and Trade-offs

This section presents several critical effects and interferences between pairs of conflicting requirements from the aforementioned requirement set in the context of P2P overlay networks. The degree at which the conflicting requirements may be fulfilled is discussed. The following list is not aimed to be complete but to raise the conflicting requirements issue.

1. **Fault-tolerance versus heterogeneity.** In the context of P2P systems where peers represent unreliable components, fault-tolerance is achieved mostly by the use of redundancy and replication mechanisms. DHT-based approaches suggest a large number of neighbors that usually increases logarithmically with respect to size of

the system. While it has been shown that such an approach provides high fault-tolerance [LBK02], it ignores practical limitations raised by peers of low physical capabilities. Moreover, it should be noted that the aforementioned study makes the assumption that peer availability follows a Poisson distribution, which does not reflect the empirically observed reality [DMS05].

2. **Scalability versus expandability.** Certain topologies where the diameter increases logarithmically with respect to the size of the network have been proposed to serve as P2P interconnection networks. However, many of these topologies can not be defined for arbitrary network sizes. For instance "perfect" hypercubes or de Bruijn networks may be defined for network sizes that increment exponentially. This limitation is critical for P2P systems where participation cannot be controlled. Certain algorithms can modify the aforementioned topologies to make them incrementally expandable, though, the resulting topologies diverge from the optimal original structures.
3. **Heterogeneity versus load-balance.** On the one hand, designing large-scale, self-organized systems may be a great challenge in scenarios where a large number of low capability and unstable peers participate. Currently, only non-autonomous systems (e.g., KaZaA, eDonkey) seem to work efficiently in wide deployments (with the exception of Overnet⁹ [Ove05]). Following such hierarchical solutions the workload is unevenly distributed among normal peers and super-peers. On the other hand, following non-hierarchical solutions needs to go beyond the currently proposed schemes to achieve efficiently fault-tolerant and stable overlays. Adaptive mechanisms are required to provide the maximum efficiency while preserving the autonomy of the peers.
4. **Anonymity versus security.** Full anonymity in every aspect of the transactions and high security levels are by themselves extremely challenging topics. Their combination is an even harder problem since in many cases designers have to decide for either alternative. In a fully anonymous environment, malicious peers can act freely while in a highly secure environment, strong identification is usually a requirement. Additionally, the identification scheme may set the level of anonymity and user privacy. Taking as an example the use of native IP addresses, actions may be mapped to users in a more direct way. On the other hand, an appropriately selected identification scheme may be combined with *cryptographic* techniques to increase the level of security in the system. It should be noted that although the proposed set of design mechanisms does not address directly security and anonymity issues they are critical requirements for many applications.

2.7. Summary

This chapter supplied the necessary requirements that have to be offered to applications and services by P2P overlay networks targeting in widely deployed, large scale, highly

⁹Overnet uses a modified version of Kademia [MM02], but no complete documentation is currently available.

dynamic and heterogeneous environments. Starting with particularly interesting application use cases, we have extracted the relevant features and characteristics of P2P systems. The applications (i.e., file sharing and CVEs) represent two of the most widely deployed P2P systems today in the Internet. Thereby, the required common functionality has been described in detailed. Additionally, the non-functional requirements that may set the success of a P2P system such as scalability, expandability, dependability and load-balance are discussed. Relevant definitions and related examples from the P2P literature have been supplied for each requirement.

Moreover, we trigger with a discussion the challenging issue of having several conflicting requirements that may not be trivially solved. High adaptation and careful design are the lines we should follow to effectively meet the complete set of them. The degree these requirements are met by existing solutions is the focus of the following chapter.

Alternative Overlay Network Design Approaches

Wisdom begins in wonder.

SOCRATES

Conceptual outline.

In this chapter, the properties of existing alternative overlay networks for P2P systems are examined. The aim of this step is twofold. First, interesting mechanisms can be assessed and potentially employed in evolved forms adaptively and secondly, the intrinsic limitations of the existing solutions may reveal the need for designing a novel architecture. The latter is the goal of the following chapter. More specifically, a three-dimensional classification of the overlay network design space is provided in this chapter, capturing accurately the most critical design factors. Subsequently, we examine the characteristics of the most relevant to our approach P2P systems. We emphasize hybrid approaches since they are capable of addressing better the tradeoffs between the non-functional requirements. In addition, hybrids may offer additional properties not present in the original mechanisms. Thereby, the suggested design of this thesis follows hybrid solutions to address multiple relevant issues, as it will become more comprehensive in Chapter 4.

This chapter is organized as follows. Section 3.1 describes the overlay network design dimensions. Then, the important class of Distributed Hash Table based structured overlay networks is discussed in Section 3.2, by additionally providing some deeper details on important representatives. Afterwards, we focus and present some details on the design of interesting hybrid approaches in Section 3.3. The chapter is summarized in Section 3.4.

3.1. Overlay Network Design Dimensions

In order to meet the critical set of the identified (and possibly additional) non-functional requirements for the operation of the P2P overlay networks, a great variety of approaches have been proposed. Analyzing the design mechanisms that characterize the P2P overlay networks, three major design dimensions can be identified to classify the proposed systems (cf. Figure 3.1). An alternative three dimensional approach is supplied in [DGMY03].

- Overlay networks vary in their *structural* design from *tightly structured* networks such as Chord [SMK⁺01] or Pastry [RD01a] to *loosely structured* ones such as Freenet [CSWH00] or Gnutella [Gnu05b]. This design dimension is graphically depicted in the projected axis of the design space in Figure 3.1. Tightly structured (or simply *structured*) overlays continuously maintain their topology, aiming at a "perfect" structure, e.g., a hypercube or a butterfly topology. Structured topologies may require high maintenance cost especially in the presence of a high churn rate. Also, they deal uniformly with the shared objects and services provided by the system and they are unaware of their distribution, a fact that might cause a significant mismatch. Moreover, *Distributed Hash Table* (DHT) based approaches (which are the most common mechanism to build structured overlay networks) cannot support easily range queries¹⁰. Alternative investigations include several mappings of local data structures on distributed network topologies, such as tries [FV02] or modifications of traditionally used topologies such as hypercubes [SSDN02], butterflies [MNR02] and multi-butterflies [Dat02].

On the other hand, loosely structured (or simply *unstructured*) overlays do not aim to reach a predefined targeted topology, but rather they have a more "random" structure. However, it has been observed that certain connectivity policies (i.e., preferential attachment) may result in topologies described by power-law networks or networks with small-world characteristics¹¹. Unstructured topologies are typically inefficient in finding published, rare items and the embedded searching operations are in general considerably costly in terms of network overhead (most approaches use flooding or at best, selective dissemination mechanisms[LRS02]). The observed power-law topology (though it provides a graph with a small diameter¹²) distributes the communication effort unevenly and introduces potential hot spots at peers with a high degree since they become "hubs" in the resulting overlay network infrastructure. However, in scenarios where the query distribution is non-uniform (i.e., lognormal, Zipfian) unstructured networks may operate efficiently.

- Further, overlay networks may vary in the *dependency* of the peers on each other, as it is shown in the vertical axis of Figure 3.1. Approaches such as Chord or Freenet treat all of the participants equally and they are referred as *pure* or *flat*

¹⁰Range queries are queries searching not for a single item that matches a specific key but rather for a set of items, which are "close" to a description based on, e.g., metadata. Refer to the discussion in Section 2.4.

¹¹The reader may refer to Section 2.3 for further details on power-law and small-world networks.

¹²Small diameter is a desirable feature for a network topology in order to reduce the maximum number of hops required to reach any destination in the overlay.

P2P networks. On the other hand, *hierarchical* approaches such as Napster [Nap05] or eDonkey [eDo05] separate the common overlay related responsibilities and assign the majority (or all) of the tasks to a small subset of (usually) more powerful nodes only, e.g., for resource indexing. This subset of peers is usually termed as "*servers*", "*super-peers*" or "*ultra-peers*". The fault-tolerance of flat approaches is considerably higher than approaches with hierarchical structure since failures or attacks to any single peer do not have as significant consequences. However, such approaches do not deal well with the heterogeneity of the participating peers both in terms of physical capabilities and user behavior. The complexity of flat approaches is usually higher compared to the hierarchical counterparts. On the other hand, hierarchical solutions require a certain infrastructure and may be controlled easier by third parties than the non-hierarchical alternatives. The operational load is unequally balanced among the networked entities and high dependency exists among them.

- Finally, overlay networks can be designed either following *deterministic* or *probabilistic* (e.g., Bloom filters [Blo70]) approaches. The selection of the former or the latter approach may improve the accuracy or the efficiency of the P2P systems, respectively. Also, a mixture of these mechanisms may provide improved results. A characteristic example demonstrating both probabilistic and deterministic mechanisms is OceanStore [KBC⁺00]. Deterministic approaches provide repeatedly similarly consistent results (as long as there are no critical intermediate changes in the system) and the provided operations can be well predicted and their cost is upper bounded. On the other hand, probabilistic approaches tolerate some unpredictability on the provided results, aiming to operate at a much lower cost than their deterministic alternatives. Such variation is shown in the horizontal axis of the overlay network design space in Figure 3.1.

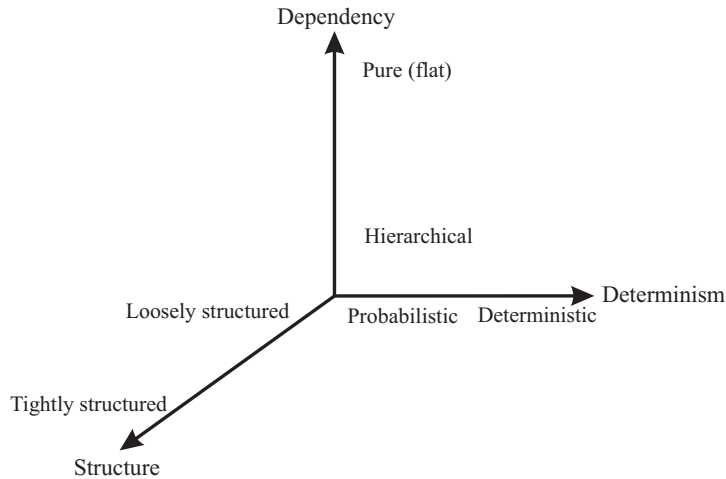


Figure 3.1.: Overlay network design dimensions.

It should be noted that several systems follow hybrid mechanisms in more than one dimensions. By doing so, hybrid designs aim to deal better with the limitations of the pure approaches.

3.2. Distributed Hash Tables

Structured overlay networks tightly control their topology and place the indexing information for the advertised resources at specified locations. Thus, subsequent queries can be routed to these locations and lookup can become more efficient. The great majority of modern structured P2P systems use *Distributed Hash Tables* (DHTs) as a communication infrastructure¹³. DHTs are powerful abstractions, where indexing information is placed deterministically at the corresponding peers with a GUID (N_{id}) closest to the data object's advertisement unique key (R_{id}). Apparently, as their name denotes, DHTs are distributed versions of the well-known hash table based data structures. DHTs use the defined hash function to select the way resources should be treated. In fact, using widely acceptable hash functions in distributed environments is a way to provide a communication mechanism without the need to exchange messages.

DHTs provide a scalable way to store and retrieve data objects under given keys [HKRZ02]. Each key lookup is resolved in multiple steps, resulting in a multi-hop path to be taken in the overlay. Thus, the core operation that is provided by DHTs is the following: *given a key, route efficiently the query to the final destination*. However, since the topologies of DHTs are usually constructed with specific constraints to provide the desired functionality, they usually do not match the topology of the underlying network.

In principle, the following steps have to be taken to design a DHT-based structured overlay network.

1. Define the key space (also called "*virtual address space*" [CF04]) and assign the keys to the participating peers and the advertised resources. It is very important that both the peers and the resources share the same key space. The length of the identifier (m) must be large enough to make the probability of keys hashing to the same identifier negligible.
2. Define the "distance" function $D(R_{id}, N_{id})$, which may be used to map the resources to the responsible peers, so that the distance function will take its minimum value. The distance function may differ considerably both in structure and semantics based on the particular system, e.g., XOR-based [MM02], prefix-based [RD01a], arc-based [SMLN⁺03], etc.
3. Define the procedure supplied to peers for populating their routing tables in order to provide highly efficient routing services. Apparently, this selection is based on the targeted topology, e.g., mesh, torus, ring, hypercube, etc. In the cases where peers have the freedom to select among a number of candidate neighbors, additional constraints or optimization policies may be applied, e.g., optimal mapping to the underlying network, observed reliability, etc.
4. Moreover, certain additional factors have to be considered, e.g., how to handle the dynamic peer participation and the resulting effects, how to redistribute the responsibilities for the address space, how to provide resilient and fault-tolerant services,

¹³Early approaches of distributed systems developed distributed structures based on Linear Hashing [LN97], [LNS96].

etc.

In the following subsections, several important DHT-based representatives are shortly described.

3.2.1. Chord

Chord [SMLN⁺03] is an important reference point in the evolution of designing DHT-based P2P overlay networks. For this reason, Chord has been extensively studied and additionally has been ported in the open architecture simulation tool for P2P overlay networks¹⁴. Pseudocode listings describing important procedures of the Chord network are provided in Appendix B.

Chord uses *consistent hashing* [KLL⁺97] to assign keys to its peers. Consistent hashing is designed to let peers enter and leave the network with minimal interruption. This decentralized scheme tends to balance the load on the system, since each peer receives roughly the same number of keys, and there is little movement of keys when peers join and leave the system. In the steady state, each peer maintains routing state information for $O(\log N)$ other peers, where N is the population of the network. Node identifiers (i.e., N_{id_i} for node i) are ordered on an identifier circle using a modulo operation (with operand 2^m), thus holding that $0 \leq N_{id_i} \leq 2^m - 1$, for each peer. In this scheme, key $R_{id} = k$ is assigned to the peer whose N_{id_i} is equal to or immediately follows k in the identifier space (assuming that there is no peer j in the system so that $k < N_{id_j} < N_{id_i}$).

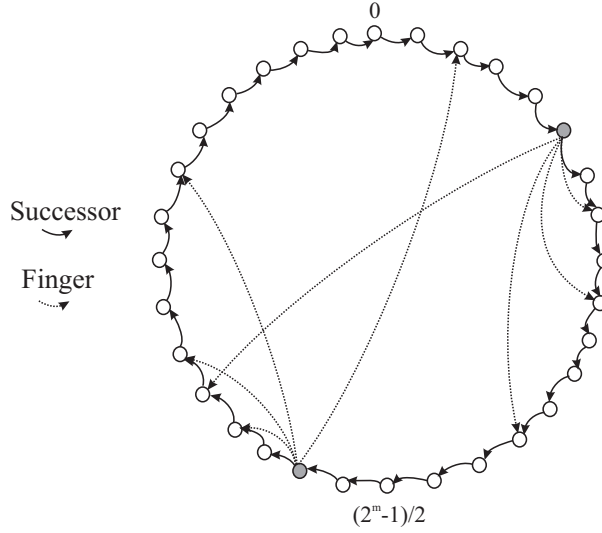


Figure 3.2.: Chord architecture.

Chord's architecture is shown in Figure 3.2 forming a ring, which is enhanced with exponentially distributed shortcuts, called "*fingers*". Each peer is connected to its successor in the ring. Maintaining correct successors is a system requirement for its correct operation. Further, while having correct successors provides valid system operation, maintaining

¹⁴The architecture of the tool is described in Chapter 7.

correct fingers (pointing to exponentially further away peers) provide efficient lookup operations. Correct fingers permit peers to halve the distance from the final destination on each routing step. Thus, on each peer join or leave action the system needs to perform $O(\log N)$ topology maintenance operations, i.e., reassigning $O(\log N)$ fingers and building $O(\log N)$ fingers for the newly joined peer. Moreover, in an improved Chord design where fault-tolerance is targeted, additional $O(\log N)$ predecessors copy their indexing information to the new peer [LNBK02].

While Chord's architecture is an interesting approach and an evolutionary step from the previously developed architectures (e.g., Gnutella [Kab01], eDonkey [eDo05], Freenet [CSWH00], etc.) by addressing the lookup scalability problem very efficiently, it does not consider every requirement found in highly dynamic and heterogeneous P2P systems. An evidence for this argument is, for instance, the fact that while it is widely accepted in the research community, it has not been deployed to any Internet-wide application. In fact, Chord deals equally with each peer, requiring each one to maintain a considerable number of fingers. More importantly, as peers remain connected with the system for short time periods only, the triggered maintenance mechanisms generate considerable traffic. Some applications have been constructed on top of Chord such as *Cooperative File System* (CFS) [DKK⁺01].

3.2.2. de Bruijn Digraph-based DHTs

Since de Bruijn graphs are used in the design of our approach, some interesting alternative approaches are studied here. Examples are systems such as *Koorde* [KK03], *D2B* [FG03b], *Optimal Diameter Routing Infrastructure* (ODRI) [LKR03] and "*Half-Life*"¹⁵ [NW03]. Moreover, Rajaraman et al. [RRV01] proposed a hierarchical, cluster-based approach which uses de Bruijn graphs to organize the intra-cluster communication scheme. Koorde and D2B are discussed in more detail.

Koorde is a proposal that deploys the Chord design over de Bruijn digraphs. The authors suggest the construction of de Bruijn digraphs with node degree proportional to the logarithmic size of the network to avoid the robustness limitations of constant degree connectivity. This requires a good estimation of the size of the network and it obligates the most attractive feature of the de Bruijn digraphs (which is the combination of having logarithmic diameter and constant node degree). Koorde suggests the introduction of "imaginary nodes" to address the incremental extendability limitation of the de Bruijn graphs.

D2B is a content addressable network that employs de Bruijn graphs to construct its overlay network. Although the proposed topology is a variation of de Bruijn graphs, they provide an interesting graph operation analysis. In D2B a procedure is suggested that allows nodes to have variable length identifiers of more than one symbol. This is even the case for linked neighbors. The resulting digraph is not always a de Bruijn one.

¹⁵In fact, the suggested system has not been given a name by its authors, thus, we provide a meaningfully descriptive name here.

3.2.3. Further Approaches

In this section, a short description of alternative P2P overlay networks is supplied to form a complete picture of the design space.

Pastry [RD01a] makes use of Plaxton-like prefix routing, to build a self-organizing decentralized overlay network. Plaxton et al. [PRR97] proposes a distributed data structure, known as the "*Plaxton mesh*", optimized to support a network overlay for locating named data objects which are connected to one root peer.

Tapestry [ZHS⁺04] has similar properties as Pastry. It employs decentralized randomness to achieve both load distribution and routing locality. In contrast to Pastry, Tapestry uses a suffix-based routing mechanism. Moreover, the handling of network locality and data object replication is performed in a different way. The architecture of Tapestry improves the Plaxton mesh structure with additional mechanisms to provide availability, scalability, and adaptation in the presence of failures and attacks (multiple roots for each data object are used by Tapestry to avoid single point of failure).

Kademlia [MM02] is symmetrical DHT-based overlay that uses a XOR-based distance metric to construct its topology and assign the resource advertisements to peers. Kademlia's symmetrical architecture enables the usage of query messages for maintenance purposes, thus, reducing the required out-of-band maintenance signalling. Kademlia allows peers to select their neighbors from sets of peers sharing the same prefix. Kademlia, Pastry and Tapestry have operation complexity comparable to that achieved by Chord.

The *Content Addressable Network (CAN)* [RFH⁺01] is a distributed decentralized P2P infrastructure. The architectural design is a virtual multi-dimensional Cartesian coordinate space on a multi-torus (d -dimensional coordinate space). The entire coordinate space is dynamically partitioned among all the peers (N number of peers) in the system such that every peer possesses its individual, distinct zone within the overall space. Each peer maintains $O(d)$ neighbors and the lookup procedure requires $O(d\sqrt[d]{N})$ steps.

SkipNet [HJS⁺03] and *SkipGraph* [AS03] are two very similar structured overlay networks (though they have been developed independently) that extend skip lists [Pug90], a probabilistic data structure. While they are similar to Chord, their basic difference is that they release the requirement that fingers must be exponentially distributed. SkipNet and SkipGraph permit peers to have fingers that are randomly located shortcuts.

Viceroy [MNR02] is a structured network based on the butterfly topology. Viceroy requires only a constant number of neighbors with high probability while its diameter is growing up logarithmically. Though, the construction and maintenance procedures are relatively complex.

AGILE (Adaptive, Group-of-Interest-based Lookup Engine) [MS03a] is a DHT-based structured overlay network that invests on the human interests to design an efficient system. As its name suggests, AGILE clusters peers based on their interests (Group of Interest - GoI). GoI are also discussed in [SMZ02] and are investigated in the context of unstructured networks in [RKP02].

As it can be observed, the design of P2P overlay networks attracted a great interest

from the research community. The list can be extended (though not exhaustively) to include *Kelips* [GBL⁺03], *Warp* [JP03], *AntHill* [BMM02], *P-Grid* [ACMD⁺03], *HyperCup* [SSDN02], *Coral* [FM03], and *pSearch* [TXM02]. Moreover, several interesting surveys provide comparisons among most of the well-known systems (cf. [ATS04], [LCP⁺04], [CF04], [HAY⁺05]).

3.3. Hybrid Topologies

The aforementioned design approaches are mostly focusing on accomplishing particular goals, e.g., scalable routing. Their common design characteristic is that they target in developing a single mechanism to reach their goal. Most of them can be clearly located at a border point on each axes of the design space in Figure 3.1. Doing so, they have limited abilities to successfully address the complex environment of large-scale, dynamic and heterogeneous P2P systems. Apparently, it is beneficial to follow *hybrid* solutions in dealing with the large number of the identified requirements. In fact, the solution suggested in this thesis combines a number of mechanisms to achieve this goal resulting in a hybrid solution. In this section we explore other existing hybrid approaches. Though initially hybrid P2P systems were targeted at merging together the P2P and C/S paradigms in different services (cf. KaZaA [LRW03], [LKR04] or eDonkey [eDo05]) they have been extended to explore a much wider range of combinations.

3.3.1. JXTA

*JXTA*¹⁶ [TAA⁺03] defines a common set of protocols for building P2P applications to address the recurrent problem with existing P2P systems of creating incompatible protocols. The main goal of JXTA is to define a generic P2P network overlay, which may be used to implement a wide variety of P2P applications and services. While JXTA offers the means to developers to design any kind of overlay network suitable to the needs of their applications, JXTA itself develops a hybrid overlay network to orchestrate the deployed applications and services. Peers in the JXTA network are self-organized into *peergroups*. A peergroup represents an ad hoc set of peers that have a common set of interests, and have agreed upon a common set of policies (membership, routing, searching, etc). However, there is a global peergroup as a bootstrap point where all the specialized peergroups can be advertised.

The JXTA specifications define the Resolver Service Protocol as a universal resource binding service. The Resolver Service is used to perform resolution operations found in traditional distributed systems, such as resolving a peer name into an IP address (DNS) or binding an IP socket to a port.

The global JXTA overlay network provides a default resolver service based on *rendezvous* peers. Rendezvous peers are peers that have agreed to index other peer advertisements

¹⁶The name of JXTA come from the verb *juxtapose*, which means place things side by side to suggest a link together or emphasize the contrast between them.

to facilitate the discovery of resources in a peergroup. A peergroup can have as many rendezvous peers as required to support the size of the peergroup. Rendezvous peers are defined in the scope of peergroups to reduce the communication complexity. Any peer can potentially become a rendezvous peer, unless there are security restrictions.

Rendezvous maintain an index of advertisements published by edge peers via the *Shared Resource Distributed Index (SRDI)* service. Edge peers use SRDI to push advertisement indices to their rendezvous when new advertisements are published. The rendezvous/edge peer hierarchy allows resolver queries to be propagated between rendezvous only, significantly reducing the amount of peers that need to be searched when looking for an advertisement. Such a structure is illustrated in Figure 3.3.

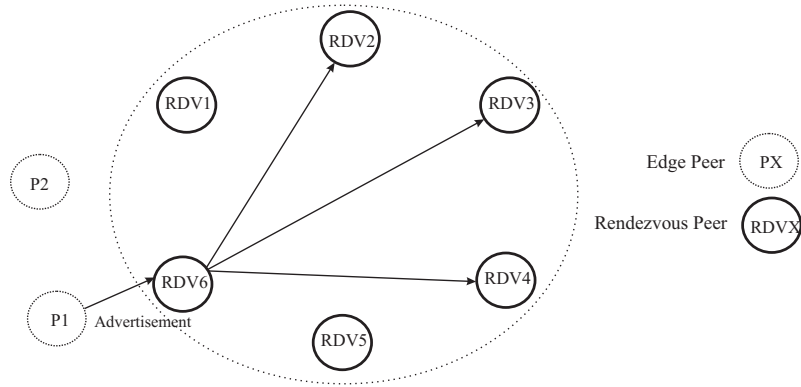


Figure 3.3.: JXTA overlay network.

Rendezvous Peers are organized into a loosely-coupled network to reduce the high maintenance cost that occurs in highly dynamic P2P systems. The JXTA approach separates the cost of a DHT solution into index maintenance, and data access. It utilizes a hybrid approach that combines the use of a loosely-consistent DHT with a limited-range rendezvous walker. Rendezvous peers are not required to maintain a consistent view of the distributed hash index leading to the term loosely-consistent DHT. Each rendezvous maintains its own *Rendezvous Peer View* (RPV), which is an ordered list of known rendezvous in the peergroup. However, inconsistency among the RPVs of different rendezvous peers may occur. A loosely-coupled algorithm is used for converging local RPV, where a rumor-based technique is employed to disseminate information about the rendezvous peers. *Seeding* rendezvous are special rendezvous peers, which accelerate the RPV convergence, as all rendezvous should know about all seeding rendezvous of a peergroup. However, as it is shown in [Pot05], large peergroups requiring several hundreds of rendezvous suffer significantly from the resulting inconsistency, which becomes a boomerang to the performance of the system.

3.3.2. Brocade

The majority of DHTs assume that most nodes in the system are uniform in resources such as network bandwidth and storage. As a result, messages are often routed across

multiple *autonomous systems (AS)* and administrative domains before reaching their destinations.

Brocade is a hybrid overlay network proposal, where a secondary overlay is layered on top of a primary DHT. The secondary overlay exploits knowledge of underlying network characteristics and builds a location-aware layer between "super-nodes", which are placed in critical locations in each AS of the Internet. Super-nodes are expected to be endpoints with high bandwidth and fast access to the wide-area network and act as landmarks for each network domain. Sent messages across different ASs can be delivered much faster if normal peers are associated with their nearby super-nodes that can operate as "shortcuts" to tunnel the messages towards their final destination, thus, greatly improving endpoint-to-endpoint routing distance and reducing network bandwidth usage.

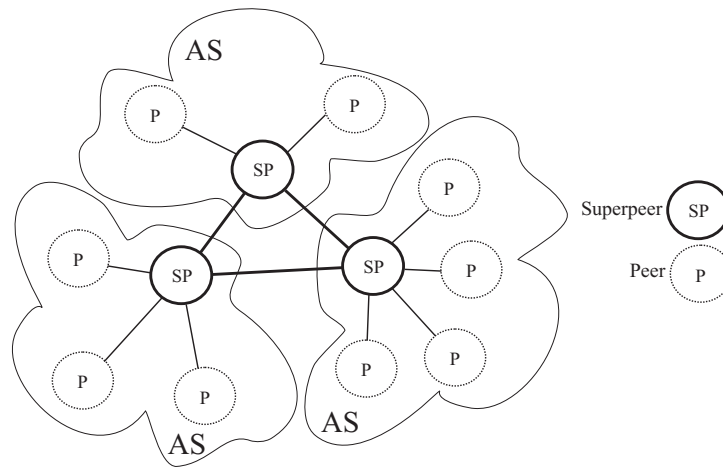


Figure 3.4.: Brocade overlay network.

The critical aspects in designing an effective Brocade overlay are the appropriate selection of super-nodes and the mappings between super-nodes and normal DHT nodes. A straightforward solution is to exploit the hierarchical structure of network domains. Each network gateway may act as a brocade routing landmark for all nodes in its subdomain. An example of this mapping is shown in Figure 3.4. Super-nodes are organized in a different DHT (i.e., Tapestry).

The routing operation works as follows. When a message reaches a super-node, the super-node can do a lookup to determine whether the message is destined for a local node, or whether brocade routing may be useful. In the brocade overlay, each super-node advertises the IDs on this list as IDs of objects it stores. When a super-node tries to route an outgoing message which should be delivered out of the local AS, it uses the super-node DHT to search for destination super-node. By finding the object on the brocade layer, the source super-node forwards the message directly to the destination super-node, which resumes normal overlay routing to the final destination.

Summarizing, Brocade is a hybrid system aiming merely to exploit the underlying network topology to supply more efficient routing services. However, the load balance of the network is unevenly distributed among the peers. Moreover, super-peers may either act

maliciously or become targets of attacks.

3.3.3. SHARK

Pure DHT-based solutions rely on hash functions, which may map advertised items to certain locations in the overlay structure (by assigning hash-generated identifiers both to each item and overlay location). Such mechanisms (while they are very efficient) are limited to single lookup queries of these identifiers. *Range* (or *rich*) search queries based on keywords remain challenging features for such systems. However, usually users prefer to specify what they are looking for in terms of keywords. For instance, a user of a file sharing application could look for a certain genre of music and not for a particular song. Additionally, multiple dimensions of meta-data are also highly desirable, for instance, looking for a document released at a certain period and related with a specific topic.

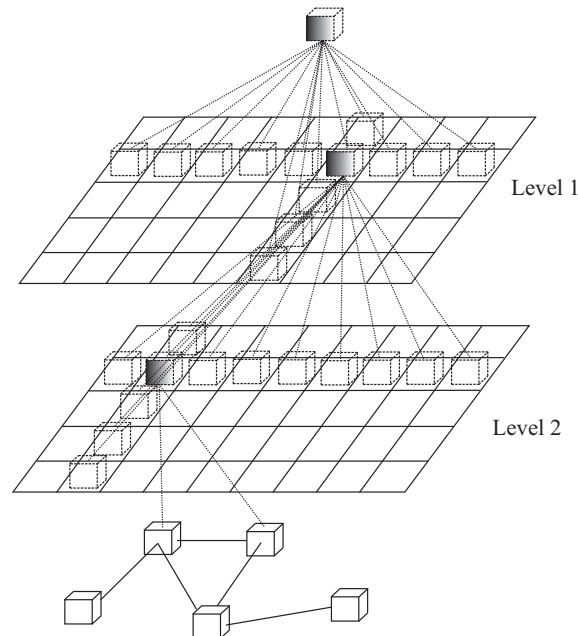


Figure 3.5.: SHARK overlay network.

SHARK (*Symmetric Hierarchy Adaption for Routing of Keywords*) [MS03b] employs a hybrid DHT solution for rich keyword searching. It extends the GoI concept of AGILE (cf. Section 3.2.3) and arranges nodes into a multidimensional Symmetric Redundant Hierarchy (SRH) scheme. The overlay topology matches the structure of the query meta-data such as to exploit the alignment for query routing. Figure 3.5 [MS03b] provides an illustration of the SHARK topology, where the descriptions are restricted to two levels and two dimensions. Each node is assigned to a GoI according to the objects it stores and to its prior request behavior. Each GoI represents a leaf in the hierarchy. SRH is a structure that provides high redundancy so that all peers have symmetric roles in the overlay.

When a SHARK node initiates or receives a query, it forwards it to all neighbors whose position meta-data exhibits a similarity greater than the predefined threshold. Further-

more, it adds information on the current level and dimension in the hierarchy that has been resolved to avoid duplication of queries. With a certain pruning probability, the requesting or currently forwarding node may already itself be on the correct position for the next hop, so that a "part of the tree can be pruned off" [MS03b] reduce the required routing effort. The forwarding process is repeated until the query reaches the corresponding leaf position in the hierarchy. From there on, it is flooded (or disseminated by means of more advanced techniques) throughout the GoI along the unstructured sub-network.

3.3.4. OceanStore

OceanStore [KBC⁺00] is a P2P storage system built on top of Tapestry [ZHS⁺04] to take advantage of its scalable characteristics. However, OceanStore, employs an additional probabilistic mechanism based on attenuated *Bloom Filters*, resulting in a hybrid solution to improve even further Tapestry's routing performance.

The Bloom Filters algorithm was initially proposed by Bloom in the early '70s [Blo70] to help word processors perform capitalization and/or hyphenation in a document. Bloom Filters exploit efficiently the usually present non-uniform distribution of requests, where a small set of items is requested much more often than the rest of the stored items. In general, Bloom Filters are capable of answering questions of the type: "Is this item member of that group"? The algorithm uses hash functions, though it requires less space and is faster than a conventional one to one hash-based mapping algorithm. However, it allows errors to happen. While negative replies to the aforementioned question are always correct (the mechanism is capable of replying correctly that an item does not belong to a group), it might provide false positive replies meaning that an item, which does not exist in a group might be falsely reported as its member. The probability of false positive replies can be configured with a number of parameters so that it is limited by certain predefined bounds. For instance, a solution is to increase the size of the filter's data structure.

OceanStore uses attenuated Bloom Filters to provide a fast probabilistic search algorithm, where attenuated Bloom Filters are arrays of such filters. In the context of the OceanStore algorithm, the first Bloom Filter (located at position '0') is a record of the objects contained locally on the current node. The i th Bloom Filter is the union of all of the Bloom Filters for all of the nodes at distance i through any path from the current node. An attenuated Bloom Filter is stored for each directed edge in the network. A query is routed along the edge whose filter indicates the presence of the object at the smallest distance.

When the fast probabilistic algorithm fails to provide the requested results, OceanStore activates the Tapestry routing mechanism to forward the request to the final destination. As a result, OceanStore provides replies much faster for the very popular items than is using a Tapestry approach. However, the routing cost is increased for the cases where Bloom Filters provide false replies. Moreover, the dissemination of Bloom Filters may consume considerable bandwidth¹⁷.

¹⁷Apparently, we address this problem in the context of our work, by proposing a caching mechanism (cf. Chapter 5).

3.3.5. Hybrid PIER

PIER [HHL⁺03] is a distributed query engine built on top of CAN. Similarly to the goal of OceanStore, in order to exploit the advantages of looking for popular items, a hybrid system has been proposed for PIER [LHSH04]. *Hybrid PIER* benefits both from DHTs and popularity-aware mechanisms, which are employed to get an improved overlay network. Hybrid PIER overlay is composed of two components: (i) an UltraPeer-based Gnutella network¹⁸ and (ii), a structured CAN where only UltraPeers may participate. The hybrid search infrastructure utilizes selective publishing techniques that identify and publish only rare items into the DHT (a decision taken by the UltraPeers). The search algorithm uses flooding techniques for locating popular items, and structured (DHT) search techniques for locating rare items.

As long as the distribution of object replicas in the system follows a long tail distribution, such a hybrid system performs better than a pure DHT alternative. However, the indexing and routing load is not evenly distributed.

3.3.6. Further Hybrid Mechanisms

In order to demonstrate the benefits of exploring hybrid mechanisms (though not directly related to our work) several hybrid approaches are further listed, showing the interest of the research community to this direction. A hybrid topology inspired by P2P overlays and applied in mobile ad hoc networks can be found in [KW04]. Further, a two-tier hierarchical Chord is explored in [GEBF⁺03]. A hybrid topology that extends Chord to increase the degree of user anonymity can be found in [SL04]. Moreover, an early comparison of some pioneering hybrid approaches such as Napster and Pointera is provided in [YGM01]. Also, a hybrid protocol named Borg [ZH03] aims in scalable Application-level Multicast, while a generic mechanism for the construction and maintenance of super-peer based overlay networks is proposed in [Mon04]. A P2P overlay network simulator has been implemented especially to augment the evaluation of a wide range of hybrid designing methods [DMLS04]. YAPPERS [GSGM03] is another interesting overlay network that combines DHT concepts with unstructured networks. Lastly, a hybrid approach on deploying hybrid Content Delivery Networks (CDNs) based on an ad hoc P2P overlay and a centralized infrastructure is described in [XCRK03].

3.4. Summary

A large number of important P2P systems (based both on pure and hybrid approaches) has been examined in order to reveal their advantages and limitations, as well as the design purpose they serve.

In summary, with respect to the non-functional requirements, the majority of the described systems address the scalable routing issue. Additionally, fault-tolerance is a major issue

¹⁸Based on Gnutella v0.6 protocol.

addressed by several existing systems. However, the way fault-tolerance is tackled is not considering in most of the cases the large amount of generated traffic. On the other hand, it has been shown that loosely structured approaches result to highly inconsistent systems. Further, supporting heterogeneity in peers' physical capabilities and user behavior is another important requirement that is related with the load balance. Most systems either ignore the heterogeneity and deal equally with every peer or define a clear separation of the roles in the system, resulting to a highly unbalanced system (which is vulnerable to attacks and with potential performance bottlenecks).

Hybrid P2P systems are interesting alternatives to pure system designs since they can overcome the limitations of the original approaches. Exploitation has shown that hybrid systems are usually the ones that are widely deployed and extensively used. It is the purpose of the following chapter to explore a well balanced architecture addressing the complete set of the identified requirements.

Overlay Network Architecture

In the world of knowledge the idea of the good appears last of all, and is seen only with effort.

SOCRATES

Conceptual outline.

Up to this point the overlay network design space has been studied and the degree at which existing solutions can address the identified requirements has been investigated. It has become clear that large-scale, dynamic and heterogeneous systems are very demanding arrangements and designing an adequate overlay network that meets the aforementioned requirements is a challenging process. In this chapter our design approach (applied in a novel overlay network called Omicron) is described. While most of the alternative proposed approaches aim to address mostly scalability and performance requirements, Omicron is particularly sensitive to offer a balanced solution by (additionally) addressing equally the intrinsic instability of the network topology (caused by the dynamic peer participation) as well as the inherent peer heterogeneity (both in user behavior and physical capabilities). Effective mechanisms and algorithms are devised to improve the adequacy of de Bruijn digraphs for P2P systems and overcome their shortcomings. The architectural description of the solution is the main focus of this chapter, while a deeper consideration of individual details as well as the modeling, analysis and performance evaluation of the system are mostly the focus of Chapters 5, 6 and 8.

This chapter is organized as follows. Section 4.1 motivates the architecture introduced in this dissertation. Then, the core components of the architecture are described in Section 4.2 and the basic mechanisms are introduced in Section 4.3. Subsequently, the architecture of Omicron is presented in Section 4.4 and the network management procedures are described in Section 4.5. Finally, the chapter is summarized in Section 4.6.

4.1. Motivation

The different existing overlay network design approaches described in Chapter 3 fail to fulfill the complete set of requirements addressed in Chapter 2 for large-scale, dynamic and heterogeneous P2P systems. The approach introduced in this thesis aims at providing both a novel overlay network *architecture* as well as several *algorithms* and *mechanisms* that can be utilized in a wide range of structured or hybrid approaches. The rationale behind the presented approach is to reduce the high maintenance cost by having a small, constant number of connections and routing table sizes (at least for the majority of the peers). For this reason, the usage of appropriate graph structures such as *de Bruijn* is chosen. These graphs are also called lexicographic¹⁹. However, despite the well-known and desirable characteristics of de Bruijn digraphs²⁰ and their wide acceptance in the P2P research community, their selection introduces a number of issues, such as *non-incremental expandability*, *potential instability of the network topology*, *larger average distance than many other alternative graphs* and *variability of node identifiers*.

Aiming at providing solutions to overcome the aforementioned shortcomings and efficiently utilize the node density of de Bruijn graphs, appropriate algorithms and mechanisms are introduced. In particular, the exponential expandability is addressed by developing an algorithm that allows the network to form an "*incomplete*" de Bruijn graph at each node addition, while still achieving the "perfect" graph structure at exponentially increasing steps. Further, the topology instability issue (caused by the highly unreliable peers) can be well addressed by introducing the concept of *endurable clusters* of peers. Endurable clusters supply stable components to construct fault-tolerant de Bruijn graphs providing the means to tolerate the *unreliable nature* of individual peers. Furthermore, the clustering mechanism results in smaller de Bruijn graphs, thus, decreasing both the diameter and the average node distance. Finally, a *dual identification scheme* is developed, where the de Bruijn node labels are assigned to clusters, while peers are assigned persistent GUIDs.

Therefore, the shortcomings of de Bruijn graphs may be effectively tackled. The suggested topology is a *two-tier* structure where the inter-cluster structural organization is designed as an incrementally expandable de Bruijn graph. However, aiming at addressing additional overlay network requirements, a role-based scheme is introduced to deal with the heterogeneity in peers' physical capabilities and varying user behavior. This scheme adjusts the contribution of each node to its capabilities and conditional reliability and aims to efficiently increase the provided cluster utility by providing appropriate incentives for each role. The intra-cluster organization depends on the assigned roles on each participating peer, which may be considered as a hierarchically structured community (though not acyclic).

¹⁹The family of lexicographic graphs includes an even *denser* graph structure than the de Bruijn, the so-called Kautz graphs [BSGL96]. However, Kautz graphs pose restrictions on the combination of the symbols that are used to label each node (aka the peer identifier). Thus, they are not adequate to represent peers, to whom any possible identifier from the address space may be assigned.

²⁰de Bruijn digraphs have been introduced in Section 2.3.3.

4.2. Core Components

Before the presentation of the core mechanisms for designing P2P overlay networks, it is necessary to describe the basic components that are required for their realization. The basic components are divided in two types: (i) *common components* used in every DHT realization and (ii) *supplementary components* that are additionally introduced in this thesis to provide the supplementary functionality for the comprehensive set of requirements that are targeted by our approach.

4.2.1. Common Components

1. **Identification scheme.** One of the first tasks in designing an overlay network is the selection of an appropriate *identification scheme* that may uniquely characterize the involved entities. Peers are the most crucial entities that require unique identification. However, potentially more components might need identification, such as clusters, shared resources, services, etc. Usually the identifiers are randomly chosen from a large key space (address space). This selection may have a great impact on other design requirements, e.g., on the efficient routing mechanism, the evenly distributed workload, the accomplished security level or the supplied anonymity features. Advanced mechanisms might be required for the identification scheme to support the notion of strong or weak identities or even anonymity in a totally decentralized way. Nevertheless, *hash-based* techniques are the most usual methods for generating global identifiers with a very high probability of uniqueness in a distributed way (without the need for a central authority).
2. **Routing tables.** The *routing tables* are vital components of the routing scheme. They should be efficiently structured and able to effectively represent peer's local view of the complete graph topology. Also, they should be sufficiently powerful to augment the efficient routing of every message independently of the destination despite the constraints of the limited local view. Constant size routing tables are preferable since their maintenance cost is not getting increased with the size of the system.
3. **Indexing structures.** In some cases the mistaken view can be encountered that structured P2P systems are designed to copy the advertised content and services at their system-specific location. Nevertheless, structured P2P systems are merely designed to hold indexing information about the actual location of the advertised items. The structure of the employed indexing scheme is very crucial for the performance of the P2P system. Indices have to be designed in a way that allows for their re-distribution and re-organization at low cost. In fact, peers may join and leave the system dynamically so there may be frequent re-arrangements of the indexed and indexing content. For example, Chord is utilizing the *consistent hashing* mechanism [KLL⁺97] to accomplish this goal at low cost.

4.2.2. Supplementary Components

4. **ClusterMap.** *ClusterMap* is a newly introduced concept employed to summarize and describe the peers participating in a cluster²¹. ClusterMaps may be realized as tables collecting entries for each member peer. Every entry may hold information about other peers' GUID, its role in the system, the observed reliability and other useful information that could be used by each peer to construct its local routing table. In fact, ClusterMaps are supersets of Routing Tables, including the potential peers of clusters that may become neighbors of a particular peer. ClusterMaps are periodically disseminated to neighbor clusters as well as the cluster itself.
5. **Roles.** *Roles* have been already introduced in hierarchical and hybrid systems to deal with the intrinsic heterogeneity of peers. However, they merely supply a C/S model by identifying two roles: (i) that of a "*superpeer*" assigned the larger part or the complete set of responsibilities (equivalent to a server in the C/S paradigm) and (ii) that of a normal peer that usually has no responsibilities (equivalent to a client in the C/S paradigm). The concept of roles can be significantly improved in order to provide a better-balanced solution that can make optimal utilization of the available resource and in parallel minimize the required operation cost; such a solution is developed in this thesis. Role assignment shall be dynamic and efficiency-driven.
6. **Rules.** *Rules* can be a useful concept to improve the efficient collaboration of the system without requiring a costly, detailed micro-payment system that accounts for the contribution of each peer, while aiming to supply a well-balanced system. Simple rules may achieve the same goal without requiring the maintenance of a demanding accounting system (which is inappropriate for low level operations provided by structured P2P overlay networks, e.g., routing, indexing). Rules have to be embedded and followed in each peer, thus achieving an effective collaboration in the absence of a central system component.
7. **Cache.** Caching is not usually found in the supplied functionality of existing tightly structured P2P systems. The main reason is that initially designers had assumed a set of requests with a uniformly distributed popularity. Though this might be a valid assumption for particular cases, the majority of empirical observations of widely deployed file sharing applications revealed the opposite. Therefore, these findings are considered in the design of our approach and an adaptive and effective caching mechanism is introduced. *Caches* have limited size and the optimal size value depends on the query popularity distribution and the observed peer behavior. It may be safely assumed that the low requirements of the caching mechanism are met by the great majority of the participating peers.

4.3. Basic Mechanisms

In this section, the description of the core mechanisms employed in our approach is supplied. First, the clustering mechanism is presented followed by the entity identification

²¹Thus, this concept is required only in systems employing peer clustering mechanisms.

scheme. Subsequently, the basic peer roles are identified (based on their provided functionality) and finally the resulting two-tier routing topology is described.

4.3.1. Clustering

Clusters have been introduced into the design of P2P systems in a variety of approaches. JXTA defines the concept of PeerGroups [Gon02] to provide service compatibility and to decompose the large number of peers into more manageable groups. Moreover, AGILE [MS03a] and SHARK [MS03b] cluster peers based on the common interests of users. Also, Considine [Con02] proposes multiple cluster-based overlays for Chord. The cluster construction in the latter proposal is based on network proximity metrics aiming to reduce the end-to-end latency. Furthermore, even hierarchical approaches like eDonkey and KaZaA might be considered as clustering approaches²² to a certain extent, where normal peers are clustered around the super-peers. The purpose of this "clustering" is to transform the costly all-to-all communication pattern into a more efficient scheme. However, by doing so it introduces additional load-balancing concerns. In fact, this is a more general issue that appears in every acyclic hierarchical organization. Thus, the cluster organization must be restricted to non acyclic structures in order to provide even responsibilities distribution.

A desirable property of each overlay network is to have a topology that remains as *stable* as possible over the time and minimize the related required communication cost to maintain the ideal structure. However, in highly dynamic P2P systems that consist of *unreliable* peers, the desirable stability cannot be attained. This is the most crucial motivating factor for introducing the concept of clusters in the architectural design of the Omicron overlay network. Clusters can be considered as an essential abstraction, which can be used to absorb the high peer attrition rate and accomplish high network stability. They can be considered as an equivalent mechanism to the suspensions used in vehicles to absorb shocks from the terrain. In order to make more clear the involved concepts, it is required to define *peer reliability*, *network stability* and *cluster endurance*. We define network stability as follows:

Definition 4.1. *Network stability $S_N(t)$ is the probability that the topology of the network remains unmodified until some time t .*

An equivalent peer reliability definition to the general Definition 2.5 provided in Chapter 2 is given below.

Definition 4.2. *Peer reliability $R_P(t)$ is the probability that the peer survives until some time t .*

Assuming that the lifespan of a peer is modeled with the random variable X , then the reliability of the peer is given by:

$$R_P(t) = \Pr\{X > t\} = 1 - F(t). \quad (4.1)$$

where $F(t)$ is the CDF of peers' lifetime.

²²The calculated clustering coefficient for these networks as defined in Equation 2.4 is relatively high.

On the other hand, a cluster is a virtual entity composed by several peers. We define the endurance of a cluster as follows:

Definition 4.3. *Weak cluster endurance $E_{CW}(t)$ is the probability that at least one peer of the cluster will survive until some time t .*

Definition 4.4. *Strong cluster endurance $E_{CS}(t)$ is the probability that at least one peer assigned a critical role of the cluster will survive until some time t .*

The distinction between the two definitions is mandatory when peers assigned non-critical roles cannot provide sufficient information about the responsibilities assigned to an individual cluster. For instance, if only a peer, which is assigned only with the Router role survives after some time t it is not possible to supply replies on queries targeted to this cluster. Though, this peer is sufficiently capable either to trigger a healing process so that peers from neighbor cluster migrate to the critical cluster or to initiate a cluster merging process. Nevertheless, it should first update the self-assigned roles to include the Maintainer role. The endurance of clusters is calculated by the following equations.

$$E_{CW}(t) = 1 - \prod_{i=1}^K (1 - F_i(t)), \quad (4.2)$$

$$E_{CS}(t) = 1 - \prod_{i=1}^{K_c} (1 - F_i(t)), \quad (4.3)$$

where K is the size of the cluster, K_c is the number of peers with critical roles and $F_i(t)$ is the CDF of the i th peer in the cluster.

The concept of conditional reliability introduced later on in Chapter 6 can provide a more accurate estimation of clusters' endurance.

Clustering algorithms aim mainly at "*partitioning items into dissimilar groups of similar items*". They require the definition of a *metric* to estimate the similarity of the items in order to perform the partitioning procedure. Since an overlay network is a virtual network, there is a lot of freedom in defining the optimal partitioning metric. In the context of P2P overlay networks, the similarity of the peers forming an individual cluster is that all the members are responsible for the same part of the *address space*, which does not necessarily define a meaningful metric for assigning peers to clusters. Therefore, the proposed clustering algorithm requires criteria other than the usual similarity metrics. The key factors that motivate the construction/deconstruction of clusters are the following:

- Clusters should fulfill the endurance requirements.
- Clusters should have the smallest possible size in order to reduce the intra-cluster communication complexity.
- Clusters should be divided when it is possible to create d other endurable clusters, where d is the degree of the employed de Bruijn graph. Selective division should be applied to maximize the endurance of each new cluster.

- Clusters should be merged when their estimated endurance is lower than a predefined endurance threshold.

Moreover, a hysteresis-based mechanism is required to avoid oscillations in splitting and merging clusters. In order to describe the membership of peers in the clusters the ClusterMap concept is employed.

4.3.2. Entity Identification Scheme

A peer identification scheme is required in P2P systems to enable the establishment of connections and provide the requested services. Further, in overlay networks based on clusters, there is a need to identify the cluster themselves. Therefore, a *dual* identification scheme is proposed to satisfy the identification requirements of the presented approach.

Peers are distinguished by a (GUID) that is created using secure hash functions²³. Such peer GUIDs have *constant* length. Clusters are distinguished by *dynamically* modified GUIDs that follow a de Bruijn-like value assignment. The peculiarity of this scheme is that the length of the identifiers is adapted to the size of the graph. Moreover, the length of de Bruijn identifiers may differ by maximum one for neighbor nodes in order to fulfill the requirement of incremental expandability and of evenly distributed workload.

The benefits of this dual identification scheme are multi-fold. Nodes can be uniquely identified and their actions may be traced when this is desirable. Thus, peers are responsible for their actions. In addition, peers cannot "force" responsibilities for indexing certain items since the mapping is not depending on the peer GUID but on the cluster GUID. Further, neighbor selection is not strictly defined. Thereby, peers can select their neighbors from neighbor clusters. This selection may be based on network proximity, trust or other specific metrics.

4.3.3. Roles

One of the major goals of this work is to provide the mechanisms to support harmonic and fair cooperation among heterogeneous peers. The first step towards this direction is the introduction of peer clusters into the architecture that share the cluster responsibilities. A naive approach that may handle heterogeneity is to assign weights to each node of the cluster. The assigned weights representing their relative abilities can augment directing the requests to the peers of each cluster proportionally (with respect to the assigned weights). While this could be helpful to handle the routing effort it would still have high requirements on each peer since all of them should to some extent participate in every core service and operation. Moreover, micro-payment mechanisms would be required to keep track of the contribution and the usage of the system in order to provide incentives for peers to declare their real capabilities.

²³For instance, MD-5 [Riv92], SHA-1 [EJ01] or similar mechanisms can be employed. Though, as computers are becoming more and more powerful, this issue should be carefully examined in a deployed system, since codes can be easier broken.

To provide support for a fair and harmonic cooperation of peers with heterogeneous capabilities and behavior, a novel approach is proposed in this thesis that aims to fit the contribution required by each peer to its physical capabilities and user behavior.

On the one hand, analyzing proposed DHT-based approaches (for example Chord, which assumes homogeneous components) it can be observed that they require every peer to participate in every core overlay operation. Peers should participate in the maintenance of the overlay, in the indexing of the advertised items, in the routing of the requests to their logical destinations, etc. Such an approach ensures a fair sharing of the effort among the peers, which, however, remain as autonomous as possible. Though, considering the given fact that peers do not have similar capabilities and that P2P systems are very unstable, it is obvious that there are big challenges in the deployment of these systems. On the other hand, hierarchical systems like eDonkey proved to be widely deployable solutions (despite the fact that they are unfair, partly centralized and with limited peer autonomy).

The main motivation behind the approach introduced here is the fact that it is too costly to design the system in a way where each peer participates in every operation. Therefore, a set of basic operations is identified that can work independently from each other. These operations are responsibilities linked to specific roles. The introduction of peer clusters is an anti-scale mechanism that augments the design of the role assignment procedure. Nevertheless, the following common basic operations/services have been identified²⁴.

- **Overlay maintenance.** *Maintainers* perform the most demanding operation since peers are required to maintain complete routing tables, indexing information as well as intra- and inter-cluster organization.
- **Indexing.** *Indexers* are required to handle the indexing responsibilities of the whole cluster in an evenly balanced and co-operative way. Information redundancy is an additional requirement in order to shield the system against the (mis-)behavior of individual peers. Indexers may provide the final reply to queries when they reach the destination cluster.
- **Routing.** Routing is the most simple operation where *Routers* should forward messages to the neighbor clusters which are closer to the final destination. Combined with the low requirements raised by the de Bruijn digraphs, the Router role is suitable for any peer, even if it has very low capabilities and an unreliable behavior. Routers participate only in the inter-cluster message forwarding service but they may not have the required information to provide the final reply as Indexers have.
- **Caching.** Although caching is not a basic operation (it is rather considered an advanced operation) it is included in the basic scheme. This decision has been taken because of the low requirements it poses and the fact that it closes the design gap between the Routers and the other more demanding roles. Caching can improve significantly the performance of the routing system [DLH⁺05]. *Cachers* are expected to perform caching of the indexing information for the most popular items in order to reduce the effort of the Indexers. Studies indicate that the observed query popularity

²⁴For particular overlay networks, e.g., designed to provide application layer multicasting services, identification of additional roles might be required.

follows a Zipfian distribution in deployed content-related P2P systems [LRW03]. Thereby, small size caches are adequate for providing replies to the majority of the queries.

It is clear that different requirements are connected with each role (since this is also the goal of the design). Table 7.1 provides a summary of the requirements for each role in an increasing requirements order. Peers are "promoted" to adopt roles with higher requirements as they prove their *reliability* and they fulfill the physical capability needs. The resulting balanced system is inspired by the successful survival of *ecological systems* where we can observe many (indirect) mutual dependencies in the food chain.

Table 4.1.: Requirements of Roles and servicing rules.

Role	Requirements			Servicing rules
	Routing table	Indexing	Reliability	
Router	Inter-cluster (de Bruijn)	No indexing		Routers Cachers Indexers Maintainers
Cacher	Inter-cluster & to Routers	Small size cache		Cachers Indexers Maintainers
Indexer	Inter-cluster & to Cachers, Routers	Cluster indices	✓	Indexers Maintainers
Maintainer	Inter-cluster, to neighbor cluster Maintainers & complete intra-cluster	Cluster indices	✓	Maintainers

4.3.4. Incentives and Rules

The questions that arises is why a peer should adopt a Maintainer role that imposes much higher requirements in terms of resource contribution and not the role of a Router that obviously requires lower contribution? In order to provide a balanced distribution of the cluster workload, a set of *rules* should be defined to give *incentives* to peers taking the roles that fit best to their capabilities and behavior. This set of rules should be both simple and powerful enough in order to be feasible and efficient. The following list provides such a simplified set of rules where the term *class* is also used to denote a role.

- Always request the service from the class with the lower requirements when it is available.
- Do not provide a service to a peer belonging to a class of lower requirements unless no other peer of such a class in the cluster can provide it (ClusterMaps can be used to check the availability of peers with different roles).
- Always provide higher priority in servicing classes of higher requirements.

These rules may regulate the load distribution among the identified classes. For example, a Router shall not forward a query to (or it shall not be served by) a Maintainer. Thereby, Maintainers are not overloaded with routing requests. Similarly, when a request reaches the destination cluster it should be routed to a Cacher initially so that Indexers will not be overloaded. If the hit in the cache is false then an Indexer will be contacted to provide the requested information and the Cache will be updated. Having reduced the workload of Maintainers and Indexers, they may mostly perform the demanding operations of maintaining the overlay and keeping the indexes updated. Additional operation related rules can be defined.

The aforementioned solution is a practical *heuristic* approach that may provide useful results. However, further analysis is required to quantify those results and provide a solution with finer granularity in fairness and efficiency that may be optimal. *Distributed Algorithmic Mechanism Design* (DAMD) techniques [FS02] are interesting alternative solutions, which may be used to formally investigate the incentives domain; however, providing such a solution is out of the scope of this thesis.

4.3.5. Two-tier Network Architecture

The suggested *two-tier* network architecture is a major step towards accomplishing the fulfillment of the targeted requirements. It enables the effective usage of de Bruijn graphs by successfully addressing their shortcomings. In fact, the successful "marriage" of two different topology design techniques (in a combination of a *tightly structured macro level* and a *loosely structured micro level*) provide a hybrid architecture with several advantages.

1. **Tightly structured macro level.** Adopting the topological characteristics of de Bruijn graphs, the macro level is *highly symmetrical* enabling *simple routing* mechanisms. Composed by endurable components it results in a relatively *stable* topology with *small diameter* and *fixed node degree*.
2. **Loosely structured micro level.** On the other hand, the micro level supplies the desirable characteristics to the macro level by following a more loosely structured topology with a great degree of *freedom* in the neighbor selection. This freedom may be invested on regulating and achieving a *finer load balance*, offering an effective mechanism to handle potential *hot spots* in the network traffic. Moreover, *locality-aware* neighbor selection may be used to maximize the matching of the virtual overlay network to the underlying physical network. Finally, *redundancy* may be developed in this micro level supplying seamlessly *fault-tolerance* to the macro level.

An example hybrid topology is illustrated in Figure 4.1. The structured level is a (2, 3) de Bruijn digraph. Two nodes (representing peer clusters) are "magnified" to expose the micro level connectivity pattern between them. Two different connection types are shown: inter-cluster connections (indicated by solid lines) and intra-cluster connections (indicated by dotted lines).

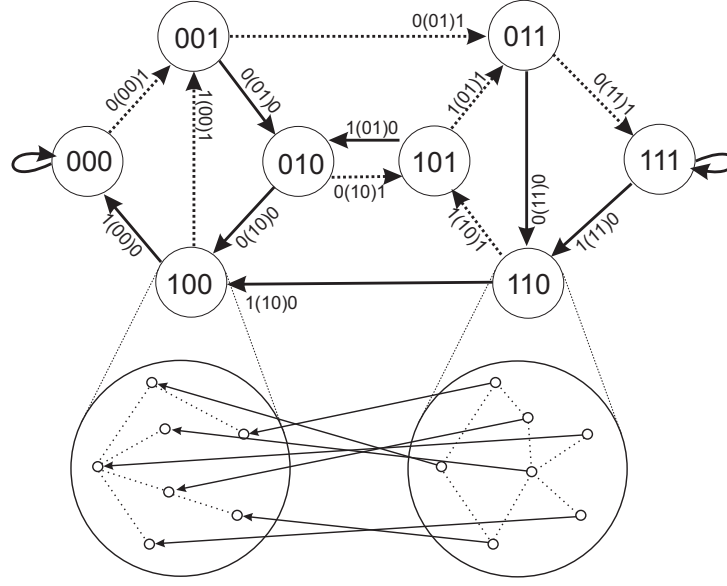


Figure 4.1.: Two-tier overlay network topology.

4.4. Omicron Architecture

Employing the suggested set of basic mechanisms and the related components a novel architecture has been developed, called Omicron²⁵. In the rest of this section a high-level description of Omicron is provided to illustrate the most important details of realizing the proposed mechanisms.

Following the proposed architecture Omicron is organized in a two-tier, hybrid approach. Peers obtain a unique identifier from a large key space and they are grouped forming enduring clusters. Clusters are uniquely identified with identifiers of variable length following the aforementioned procedure. Figure 4.2 provides a snapshot of a cluster and the connections of its peers to explain Omicron's connectivity patterns both for inter- and intra-cluster communication. Cluster identifiers are related with the left shift operation that correlates neighbor nodes in de Bruijn digraphs (L represents their common label part and C_X the most significant symbol dropped by the shift operation).

- **Roles in clusters.** The Maintainer has a central position. Peers are required to be stable to efficiently fulfill the requirements of this role. Moreover, since the requirements of the Cachers are not much higher compared to the Routers it is preferable for efficiency reasons to assign more Cachers than simple Routers.
- **Inter-cluster organization.** In this example peers are required to have a two degree inter-cluster connectivity. Maintainers of neighbor clusters exchange lists of participating peers in each cluster and this information is distributed to the peers.

²⁵Omicron is the name for the Greek letter 'O'; however, in the context of our thesis it is the acronym for *Organized Maintenance, Indexing, Caching and Routing for Overlay Networks*. It expresses the role-based organized peer communities forming enduring clusters in the proposed hybrid, two-tier overlay network architecture.

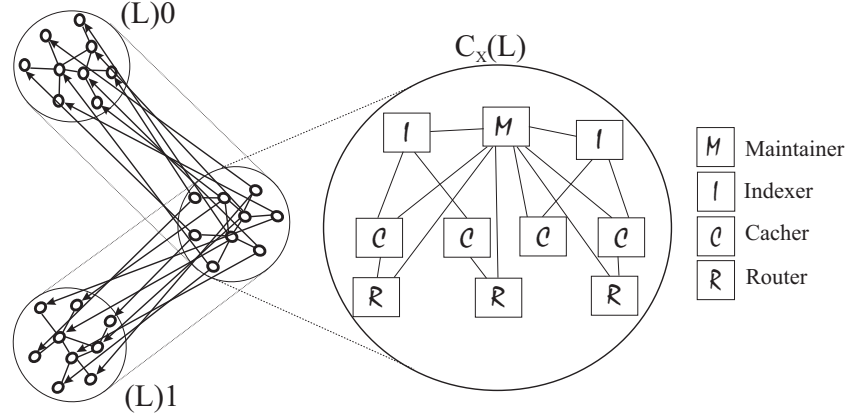


Figure 4.2.: Inter- and intra-cluster connectivity pattern.

Then each peer selects another one in the neighbor cluster to forward the messages. The mapping can be randomized or follow a certain algorithm to maximize the load-balance.

- **Intra-cluster organization.** In Figure 4.2 not all of the connections among the classes are shown to increase figure's legibility. For the same reason connections between peers of the same class have been omitted as well. However, there is a clear need for communication among Maintainers or Indexers to keep the routing tables and the indexing information updated. Here a certain level of redundancy is used to protect against failures.

4.5. Overlay Network Management

After having described the network architecture and the related mechanisms and components, the focus of this section is directed to the network management procedures. In the resulting two-tier architecture two entities are used to construct the network topology: individual peers and clusters of peer. Thereby, a set of efficient procedures to handle the dynamic participation of the peers in the system and the resulting consequences in both the endurance and the maintenance cost of clusters need to be defined. Three crucial requirements are driving the developed solutions: *dependability*, *load-balance* and *efficiency*.

4.5.1. Join Operation

A variety of bootstrap phases may provide an initial online peer P_Y that may trigger the mechanism to accept the newly joining peer P_X (cf. Section 2.4.1). Without loss of generality it can be assumed that P_Y has been assigned the Maintainer role (otherwise the request has to be simply redirected to another peer \hat{P}_Y of the same cluster that has been assigned the Maintainer role).

Upon the reception of the joining request P_Y triggers a *sampling* procedure using an inter-cluster *random walk*. It contacts a Maintainer²⁶ P_Z of a randomly selected neighbor cluster, which is recursively repeating this step making a random walk of length $w = \alpha \log(C_S)$, where C_S is the number of clusters in the system and α is a weight. It should be noted that w is asymptotically equal to the diameter of the inter-cluster overlay network. The goal of the procedure is to equally distribute the new peers in the deployed clusters considering the internal state of each cluster, i.e., its endurance and its size. By performing a random walk of length at least equal to the diameter of the network, every cluster has a probability of being included in the sampling procedure of each join request. Moreover, having a logarithmic number of samples provides a fairly good approximation of collecting the state of all clusters, which would have been very costly in terms of communication traffic. Thus, the selected approach can provide a well-balanced outcome with a low cost.

After performing the sampling procedure with the random walk, the appropriate cluster is selected. In this phase, a newly joined peer is considered unreliable and it is merely assigned the Router (and optionally the Cacher) role. Thus, the selected cluster is the one with the least number of unreliable peers so that the load for the Maintainers of the clusters is fairly equal.

The last Maintainer of the random walk indicates the newly joined peer the cluster it should become member of (via an **Accept** message). Afterwards, P_X is asking the provided maintainer of the target cluster to connect and become a cluster member. As a reply, the Maintainer provides updated ClusterMap structures of the cluster itself and the neighbor clusters so that the P_X can correctly build its routing table.

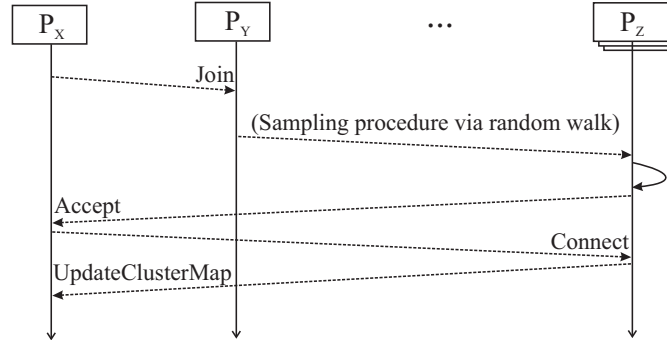


Figure 4.3.: Join sequence diagram.

The whole process is illustrated in Figure 4.3 by a sequence diagram. Peer P_Z represents the Maintainers that participate in the sampling random walk. The selected Maintainer receives the *Connect* message and replies by providing the necessary ClusterMap structures.

²⁶In the rest of the thesis a peer may be called with the highest assigned role.

4.5.2. Expandability

De Bruijn graphs have been mainly proposed for interconnection networks of parallel systems that are much more stable compared to P2P systems. Further, they are usually centrally administered. They represent excellent architectural structures for static environments, though. However, special adaptation is required to make de Bruijn graphs suitable for dynamic environments. In terms of expandability this means that while "complete" de Bruijn graphs may be defined for exponentially increased network sizes k^D , where k is the degree of the graph nodes and D the graph diameter, it is not trivial transforming them to incrementally expandable structures.

In order to address the incremental expandability of de Bruijn graphs, Fiol et al. [FL92] proposed a theoretical approach named Partial Line Digraph. Tvrdik [Tvr94] described some general algorithms that basically apply for Kautz graphs, but they could be applied in de Bruijn graphs as well. However, these techniques are merely of theoretical interest, though, their ideas may support the development of a practical approach. In fact, in the proposed approach a heuristic method is developed to deal efficiently with this problem requiring only local information. As it has already been mentioned, the nodes in the following discussion represent clusters of peers and not individual peers.

More specifically, the one and two nodes cases are trivial cases and they are complete de Bruijn graphs of diameter zero and one, respectively. Figure 4.4 pictures a de Bruijn(2, 2) graph using solid links and white nodes. The extended de Bruijn(2, 3) graph using dotted links and grey nodes can be obtained by transforming every link to a node and adding the appropriate connections among them. The identifiers of the new nodes are defined by the concatenation operation on the source of the directed link with the newly shifted bit that leads to the destination. For example, the link that connects nodes (00) and (01) takes (1) as transition bit, so the newly transformed node will get a (001) identifier.

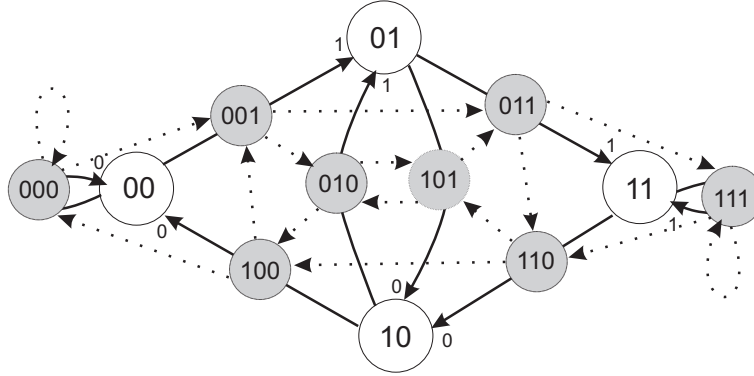


Figure 4.4.: Complete extension of a de Bruijn(2, 2) digraph to a de Bruijn(2, 3) digraph.

The incremental extension²⁷ of the original graph to the extended one is shown in a

²⁷Actually, the suggested expandability algorithm does not supply a random graph for any graph size n , $\forall n \in \mathbb{N}^*$. It rather provides graph structures with size \acute{n} , where $\acute{n} = (d - 1)n, \forall n \in \mathbb{N}^*$, which is adequate for the needs of our work.

detailed illustration in Figure 4.5. Applying the endurance-driven metric on the clusters (a local operation performed in each cluster) some of them may get a high endurance value (d times more than an appropriately defined endurance threshold). In order to keep the intra-cluster complexity as low as possible, this cluster is being split in d new clusters. The algorithm works as follows:

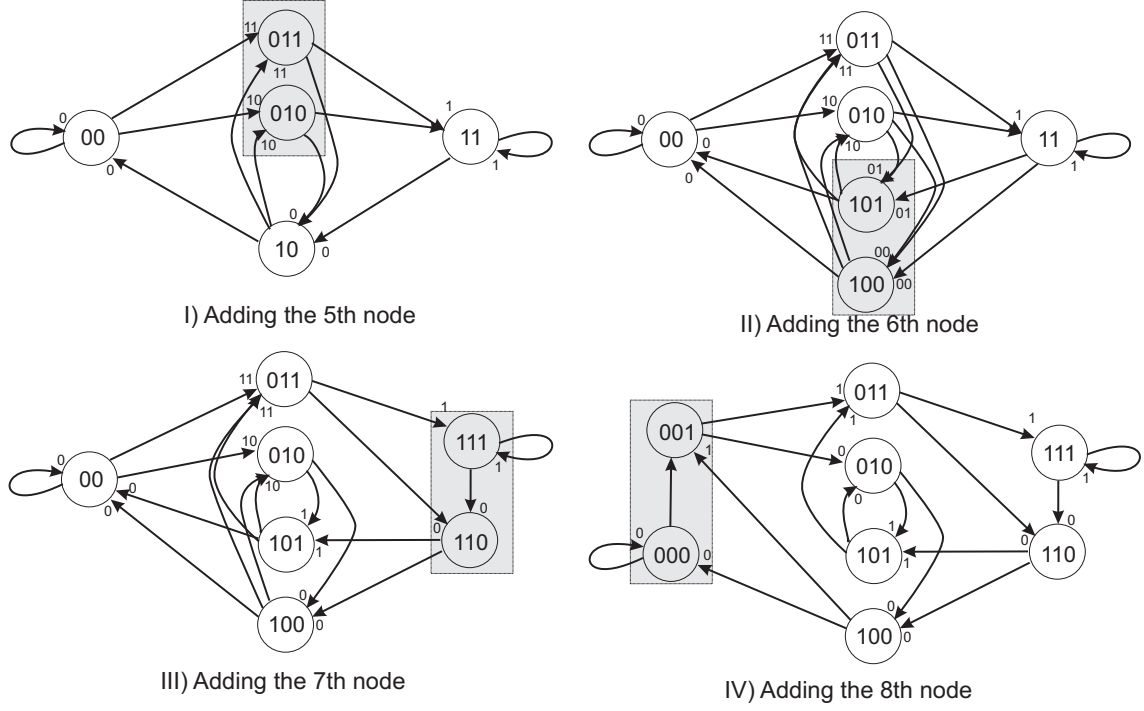


Figure 4.5.: Extending a (2, 2) de Bruijn graph, by adding nodes incrementally.

1. Select the node to split:

- If all of the neighbor clusters have the same identifier depth, then split the initially selected group in k new clusters (by increasing their identifier depth). The new clusters are constructed by concatenated the prefix of the initial cluster identifier with one additional symbol. This symbol takes one of the k distinct values $(0, k - 1)$ (different value for each new group). For example, in Figure 4.5 (I), node (01) is split into (010) and (011).
- If there is at least one neighbor cluster with shorter identifier depth then this is the cluster that must be split. Redistribute the nodes so that the cluster with the lower identifier will have excessive endurance and repeat the previous step on that group²⁸. As an example consider the case where cluster (010) fulfills the endurance requirements and it can be split. Since clusters (11) and (10) have shorter identifier length, one of them is selected to be split. The case is demonstrated in Figure 4.5 (II), where cluster (10) is selected to be split into

²⁸Actually, this should be a rather infrequent event since (as described above) the proactive peer joining procedure selects clusters with lower endurance resulting to a well-balanced system.

(101) and (100). However, in this scenario peers are migrated to cluster (10) before splitting. Further criteria can be applied in the case where more than one neighbor clusters have shorter identifier length.

2. Set new connections:

- If the newly created nodes (clusters) have longer identifier depth compared to at least one neighbor cluster, then place the same connections as the initial cluster (both for incoming and outgoing links) to all of the d newly created clusters. Incoming connections to the initial cluster are maintained for each new one. This case is shown in Figure 4.5 (I) where both nodes (011) and (010) have connections with nodes (11) and (10) as (01) did. Additionally, node (10) has connections to both (011) and (010) since initially it had a connection with (01).
- If the newly created clusters have the same identifier length as all of the neighbors apply the normal de Bruijn connectivity pattern to them and their neighbors. This case is shown in Figure 4.5 (IV) where the digraph has reached a "complete" state (2,3). It can be easily observed that all nodes have connections similar to the digraph in Figure 2.3.

3. Configure routing mechanism:

- If the neighbor of a cluster where the message should be routed to has longer identifier length (which can only differ by one), make the decision based on the two most significant bits instead of the normal operation that requires a single bit selection. As an example, consider the case of node (10) in Figure 4.5 (I). In order to route a message to either (011) or (010) it has to parse the next two symbols of the key, while in order to route to (00) it must parse only one symbol.
- If there is at least one neighbor cluster with shorter identifier length then (if the destination cluster has similar identifier depth) it should make the decision based on the two most significant bits. Otherwise it should select the next neighbor by a single bit operation and ignore its own least significant bit. For example, consider the case of node (100) in Figure 4.5 (III). In order to route a message to either (011) or (010) it should parse two symbols of the key, while to route the message to node (00), it should parse only one symbol and ignore its least significant bit.

4.5.3. Overlay Stabilization

Removing a cluster is an operation that the system is configured to avoid. This is achieved by carefully choosing the right cluster where a new peer shall join. In this phase attention has to be paid to the maintenance of the endurance of the clusters, while additionally considering the trade-off between the accomplished endurance and the cost for the join operation. As it has been mentioned above, peers can for example do a random walk in order to select the weakest cluster to join.

However, there might be cases where the evaluated cluster endurance is decreasing to a value lower than an appropriately predefined threshold. In such cases, a specific algorithm performs the reverse procedure of creating new clusters and merges neighbor clusters in order to minimize the cost of cluster "disappearance". However, a step before this action is to try migrating reliable peers from endurable neighbor clusters. Nevertheless, Figure 4.6 shows two different cases of clusters that should be removed (since it is the only available option in this scenario).

1. In the first case a cluster that has no neighbors with longer identifier depth is marked as insufficiently endurable. In this scenario, the d "sibling" clusters that share the same prefix of $(k_{max} - 1)$ length should be merged into one cluster with $(k_{max} - 1)$ identifier length. This is illustrated in Figure 4.6 (I), where cluster (101) is unstable, so it is merged with (100) to form cluster (10).
2. The second case occurs when it is required to preserve the system invariant of "neighbor cluster identification depth should differ only by one". If the insufficient endurable cluster has neighbor clusters with longer identifier depth then one of them is selected from which peers migrate to the insufficiently endurable cluster. This way the second case is transformed to that of the first case, since the insufficient endurable cluster has the longest identifier depth. This scenario is shown in Figure 4.6 (II), where instead of removing the initially insufficiently endurable cluster (11), peers from cluster (100) migrate to (11) and will result in a similar digraph like case (I).

Connections and routing tables are accordingly updated to reverse the previously described extension procedure.

Assuming sufficiently effective prediction of cluster endurance, the method introduced in this thesis addresses fault-tolerance in a different way than the commonly used high connectivity pattern. Endurable clusters are used as components in the de Bruijn overlay that have much lower join/leave rate than the single-peer based alternatives. It has been observed in many deployed P2P systems (i.e., [SGG02], [KWX01], [SW04a]) that peers which are connected for a long time tend to stay connected for longer. Taking advantage of these measurements critical and demanding roles (i.e., Indexers and Maintainers) are assigned to predictably more reliable peers. This way the commonly introduced requirement within the connectivity pattern where peers should maintain a number of connection that is a logarithmic function of the overlay network size [LBK02] is relaxed. This procedure is discussed in detail in Chapter 5.

The degree for most of the peers is *fixed*. The only exception applies to peers assigned the Maintainer role that have higher intra-cluster connectivity. Though, it is limited to the cluster size which is assumed to have reasonable upper bounds. Supplying regularly each peer with an updated ClusterMap provides multiple options to ensure intra-cluster coherence. Therefore, the increased connectivity that is applied to obtain fault-tolerance (among other desired properties) is related to the *locally observed peer unreliability* rather than the *size of the global P2P network*. In addition, the maintenance cost is better adapted to the network needs compared to the approaches that employ logarithmically growing connectivity patterns. Moreover, the structured cooperation of the Indexers and

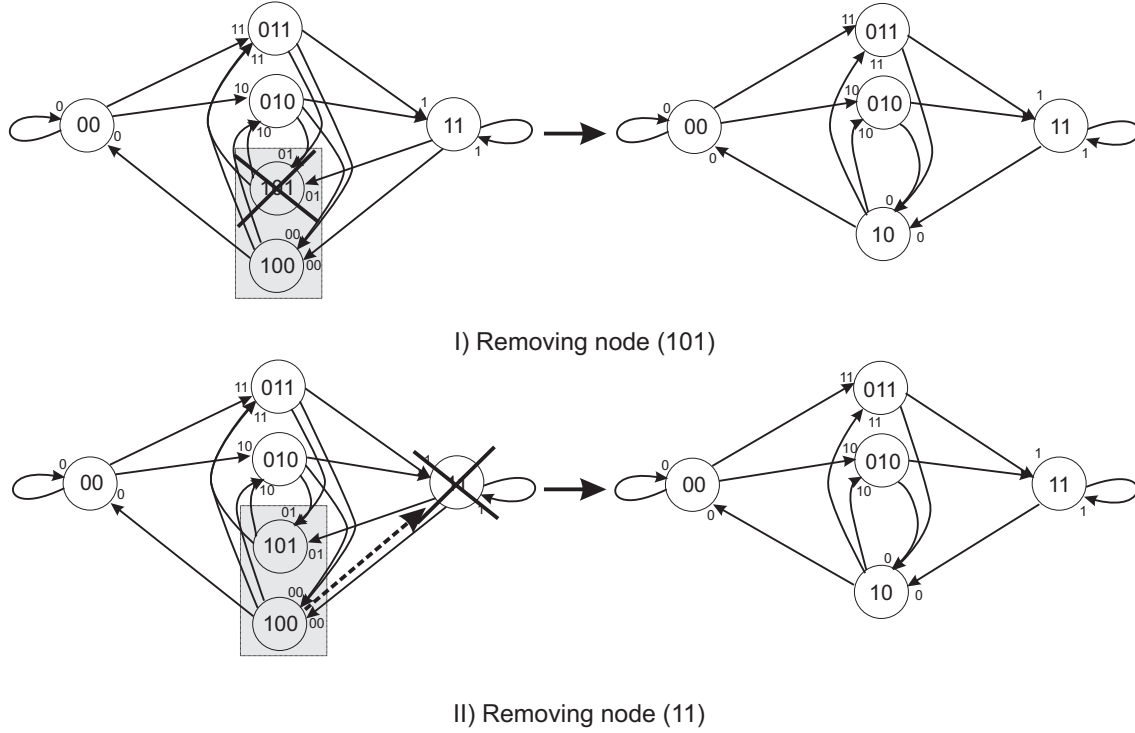


Figure 4.6.: Removing unstable clusters.

Maintainers ensures the consistency and the availability of the indexing information.

4.5.4. Enhanced de Bruijn Topology

As it can be noticed in Figure 2.3, nodes (000) and (111) are self-connected. In fact, each node of the form $(u_1u_2...u_D)$ where it holds that $(u_1 = u_2 = ... = u_D)$, is a self-connected node. Such nodes are less loaded with routing traffic since their incoming and outgoing degree is practically $D - 1$. This can be also empirically observed with simulation experiments, as it is shown in Chapter 8. In order to reduce the lower node degree effect, an enhanced de Bruijn structured has been advised. The enhanced structure connects self-connected nodes to each other. For example, the de Bruijn graph shown in Figure 2.3 is modified as shown in Figure 4.7. The dashed connections between nodes (000) and (111) can improve the routing load balance of the network. However, the routing algorithm has to be changed accordingly.

4.6. Summary

The presented network topology design and the related components, mechanisms and algorithms provide a P2P architecture capable of effectively addressing multiple non-functional

Protocols and Role Mechanisms

All things will be produced in superior quantity and quality, and with greater ease, when each man works at a single occupation, in accordance with his natural gifts, and at the right moment, without meddling with anything else.

PLATO

Conceptual outline.

The identification of roles for peers based on common, core operations is one of the key features of the overlay network architecture introduced in Chapter 4. In this chapter, the detailed description of the identified roles is provided. Their description includes the communication protocols for those roles that need to co-operate in order to supply a complete service. Additionally, the basic functionality is discussed and the related algorithms used to provide the functionality are described in pseudocode. Of particular importance is the identification of the common functionality required by many roles. Moreover, the potential co-operation between roles is investigated. In fact, roles assigned to an individual peer can communicate only with similar roles in remote peers in order to keep their design as simple as possible. Therefore, an additional default role (called Connector) is introduced in this chapter. The Connector provides the means to successfully implement the aforementioned inter-role communication constraint. Nevertheless, roles assigned to the same peer can freely provide their functionality to each other via conventional method invocation mechanisms.

This chapter is organized as follows. Section 5.1 motivates the need for the particular roles introduced in this thesis. Then, the basic protocol functionality and the inter-peer communication mechanism are discussed in Section 5.2. The description of each role is given in the following four sections. The Router is described in Section 5.3, followed by the discussion of Maintainer details in Section 5.4. Subsequently, the functionality of the Indexer and the Cacher are given in Section 5.5 and Section 5.6, respectively. Then, some common scenarios are described in Section 5.7 in order to reveal the inter-role co-operations. Finally, the chapter is summarized in Section 5.8.

5.1. Motivation

Specialization according to individual's abilities is an advantageous policy to effectively deal with heterogeneity of peers' physical capabilities and user behavior. This direction has been followed in the design of the P2P overlay network presented in this thesis, as it has been discussed in Chapter 4. Most of the identified roles need to communicate with other roles located in remote peers or locally.

Each individual role has different communication needs resulting in multiple different protocols composed by several messages. However, it is worthwhile to investigate the possible similarities that may be found in (by definition) different messages. Moreover, there is a need to enable communication between asymmetrically composed peers (assigned with different roles). Vital information should be delivered to each peer independently of the assigned roles.

Further, the deeper investigation of the identified roles can provide better understanding of the system operations. Studying the required functionality of each role, specific algorithms and mechanisms can be utilized to fulfill the requirements. Of particular importance are approaches that provide effective and low cost communication solutions. The network topology is a critical factor for selecting communication schemes that effectively deal with the utilized *hop-by-hop* paradigm.

Finally, the investigation of basic scenarios involving multiple peer roles is necessary in order to make the inter-role relationships and the way they co-operate to provide the targeted operations and services more comprehensive.

It should be noted that the goal of this chapter is not to provide a validated set of protocols that can handle every potential use case, including scenarios where peers behave maliciously. Rather, the chapter aims to provide the necessary functionality in order to demonstrate the validity of the overlay network design introduced in this thesis.

5.2. Protocol Basics and Inter-peer Communication

Several of the developed protocols follow the hop-by-hop paradigm to deliver the messages to their destination. The assumed communication mechanism uses a local message dispatcher on every peer. The message dispatcher examines several message fields in order to direct the incoming message to the appropriate instance that can handle it. Further details on the developed message dispatcher can be found in Section 7.5.

5.2.1. Common Protocol Functionality

Inter-peer communication is implemented via a number of messages that constitute the necessary communication protocols. A number of common fields appear in each defined message. Table 5.1 provides the structure of an abstract message that defines the common fields. Every concrete message is derived from this abstract message structure.

Table 5.1.: Common message structure.

Field	Description
<i>Name</i>	Name of the message.
<i>Scope</i>	Defines the related overlay network.
<i>Type</i>	Relates the message to a specific role.
<i>Style</i>	Defines the way the message is forwarded towards its destination.
<i>Initiator</i>	Encapsulates the GUID of the peer that created the message.
<i>Sender</i>	Encapsulates the GUID of the peer that forwarded the message.
<i>Destination</i>	Encapsulates the GUID of the peer that should receive the message.
<i>TTL</i>	Time-to-live (optional).
<i>Hops</i>	Number of visited peers (optional).
<i>Visited</i>	List of traversed peers (optional).

The *name* field uniquely identifies each individual message (e.g., `UpdateClusterMap`, `LookupRequest`, etc.). The *scope* of the message defines the related overlay network. For example, an inter-cluster message has a different scope compared to an intra-cluster message. The *type* of the message specifies the role that should handle the incoming message, e.g., a Router, a Maintainer, etc. The *style* of the message defines the forwarding communication style, i.e., recursive or iterative. The style field is described in further detail in Chapter 7 in the context of the developed simulation framework.

The *initiator* field encapsulates the GUID of the message originator. Supplementary information is provided in the *sender* field that encapsulates the GUID of the most recent peer that forwarded the specific message. The combination of these two fields provides the essential information to enable efficient communication schemes for query-reply based protocols. For example, a reply can be directly provided to the originator of any query without requiring to reversely traverse the same path that the query passed through. However, it should be noted that in particular cases the semantics of routing can change, e.g., involving the caching mechanism described in Section 5.6. Thus, intermediate nodes may be visited in order to deliver the reply, too. The *destination* is set to the next closest peer in every hop based on the local routing table. Then, the message is forwarded to this peer by using the underlying network.

Finally, some optional fields can provide additional useful functionality. For example, setting the *Time-to-Live* (TTL) to a maximum value can decrease the generated traffic (particularly useful when broadcasting mechanisms are utilized). However, it should be noted that the scope of the message is limited when the TTL field is actively used. The following two fields (*hops* and *visited*) provide the means to construct non-oblivious messages. Thereby, it is possible to avoid cycles in the routing procedure, which is proven to be especially useful for random walk based communication mechanisms (that require single visit of peers).

5.2.2. Inter-role Communication Model

Roles assigned to an individual peer can communicate only with similar roles in remote peers in order to keep their design as simple as possible. Moreover, since peers have different capabilities, different roles may be assigned to them. However, frequently there is a requirement that a role assigned to a peer must deliver some information to a remote peer that has not been assigned with the same role. For example, a Maintainer should provide an updated ClusterMap to a joining peer that has been assigned only with the Router role. In this case an additional mechanism is required to augment information delivery. A simple way to implement this mechanism is to assign an additional basic role to every peer, which provides the necessary message exchange, without altering the semantics of the identified roles. Such a role can fill the gap in the inter-role communication procedure.

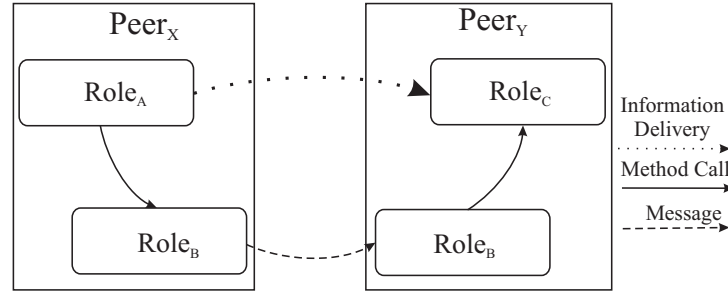


Figure 5.1.: Inter-role communication mechanism.

The concept behind this mechanism is graphically illustrated in Figure 5.1. Assume that $Role_A$ assigned to $Peer_X$ must deliver some information $I_{A \rightarrow C}$ to $Role_C$ assigned to $Peer_Y$. Moreover, $Role_B$ has been assigned both to $Peer_X$ and $Peer_Y$. Assuming that $Role_B$ has been designed in a way that it can deliver the specific information $I_{A \rightarrow C}$ between different peers, an indirect communication takes place. $Role_A$ calls $Role_B$ of $Peer_X$, which delivers the information to its equivalent role in $Peer_Y$. Then, $Role_B$ calls a local method of $Role_C$ in order to pass the information to its destination.

5.2.3. Connector

From the previous discussion, it becomes clear that a common role needs to be assigned by default to each peer. Despite the fact that the Router role is assigned to each peer, it is semantically inappropriate to be selected for this purpose. The aim of the common default role is to increase the inter-operability between peers assigned with different roles.

An important feature of this role is to offer the means to dynamically update the roles "installed" on each peer. Moreover, in terms of the adopted cluster-based approach, there is a need to provide a mechanism to dynamically update the local ClusterMaps of each

peer, as these structures evolve over time. In addition, a sufficiently rich mechanism is required to establish inter-peer connections. Two peers are considered as neighbors only if they have a mutually agreed established connection (which differs from the concept of TCP connections). Connections can be considered either as unidirectional or bidirectional based on the needs of the overlay network. Finally, peers should be able to periodically check the validity of the established connections. This can be implemented, i.e., using **Ping/Pong** messages²⁹. The following list summarizes the required functionality.

1. Dynamic establishment and tear-down of inter-peer connections.
2. Periodic checking of the validity of the established connections.
3. Dynamical installation of new roles on peers.
4. Dynamical updating of ClusterMaps on each peer.

Several messages are needed to provide the aforementioned functionality. A new peer may initiate its participation in a P2P system by submitting a **JoinRequest** message, which may be handled only by peers assigned with the Maintainer role. The first reply a new peer receives after the request to join is the **Accept** message. The **Accept** message includes a list of *PeerAddresses* that are candidate neighbors for the newly joined peer. The list may include only one peer, e.g., consider the case of Chord where only the ring successor suffices to establish the initial connection and then dynamically updates the fingers. Alternatively, it may include a list of peers, e.g., consider the case of Omicron where all the available Maintainers of the cluster may be provided to select one for the initial connection. The structure of the **Accept** message is given in Table 5.2. It extends the structure of the basic message provided in Table 5.1. In the rest of this chapter, each defined message is derived from the common message structure.

Table 5.2.: **Accept** message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>PeerAddresses</i>	Provides a list of valid peers that can become direct neighbors.

In order to install new roles at peers, the **UpdateRoles** message is used. It includes a list of the new roles to be installed and it may be initiated only by peers that are assigned with the Maintainer role.

Table 5.3.: **UpdateRoles** message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Roles</i>	Provides a list of additional roles that can be installed on the peer.

After receiving the **Accept** message, a peer requires a connection from an active Maintainer. At this initial connection phase as well as at further points in time (periodically

²⁹The **Ping** and **Pong** messages form a mechanism to actively check whether the remote party is still available. Upon the reception of a **Ping** message, peers must promptly reply with a **Pong** message.

defined or event triggered), peers receive `UpdateClusterMap` messages that include updated `ClusterMaps`. Their information is essential for the correct operation of each peer, which updates its local routing tables and the established connections with currently active peers. An `UpdateClusterMap` message may be sent only by peers assigned with the `Main-tainer` role. The structure of an `UpdateClusterMap` message is provided in Table 5.4

Table 5.4.: `UpdateClusterMap` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>ClusterMap</i>	Provides an updated <code>ClusterMap</code> that can be used to update the local routing table and the established connections.

In addition, four more messages have been defined for the `Connector` role. The `Connect` and `Disconnect` messages are used to establish and tear-down connections, respectively. The `Ping` and `Pong` messages are used to check the validity of the already established connections. Their structure is simple including the common message fields and optionally the local time a peer initiates the `Ping` message, which is copied in the `Pong` message as well.

5.3. Router

Efficient routing is the most crucial functionality expected to be provided by a network system. As it has been discussed in Chapter 3, the main aim of DHT-based structured overlay networks is to provide efficient routing. The importance of efficient routing is investigated in [RSS02] and [Gav01], where the authors raise questions on the optimal routing that can be achieved. In terms of de Bruijn graphs, which appear to be one of the most promising network topologies for P2P systems, the complexity of the routing procedure increases logarithmically with the size of the system. This is formally denoted as $O(\log(N))$, where N is the number of nodes, holds when the node degree is fixed. For cases where the node degree also increases logarithmically with respect to the network size the resulting routing complexity is discussed in [KK03]:

$$D = O\left(\frac{\log(N)}{\log(\log(N))}\right), \quad (5.1)$$

where D is the diameter of the de Bruijn digraph.

The optimal routing procedure in de Bruijn graphs has been investigated by other researchers as well. In particular, de Bruijn graphs have been studied in [MY00], aiming to combine short routing paths with fault-tolerant routing. Also, routing schemes for de Bruijn networks mostly applicable for parallel systems have been considered in [HW97], aiming to evaluate oblivious and non-oblivious mechanisms. Undirected de Bruijn graphs have been investigated in [LS96], focusing on the computational complexity of the routing algorithms. A pattern matching based approach for optimal routing is provided in [Liu90]. Finally, Sivaraman and Ramaswani provide a shortest path solution for de Bruijn digraphs

[SR94]. The simplicity of this solution makes it an interesting approach. In fact, this solution has been adopted and extended to fit the needs of our work.

5.3.1. Routing Algorithms

In order to construct a shortest-path algorithm for de Bruijn digraphs an operation `shift_match(shift, K, L)` (where $0 \leq \text{shift} \leq D$) is defined in [SR94]. In terms of Omicron, $K = (k_1 k_2 \dots k_D)$ is the GUID of the cluster that includes the particular Router peer forwarding a message. $L = (l_1 l_2 \dots l_D)$ is the GUID of the destination key. The operation on the two keys returns `true` if and only if:

$$(k_{1+\text{shift}} k_{2+\text{shift}} \dots k_D) = (l_1 l_2 \dots l_{D-\text{shift}}), \quad (5.2)$$

and `false` otherwise. The operation is described in Algorithm 5.3.1.

Algorithm 5.3.1: `SHIFT_MATCH(shift, K, L)`

```

for i ← 1 to sizeof(K) - shift
  do { if (K[i + shift] ≠ L[i])
      then
        return ( false )
    }
return ( true )

```

In addition, it is necessary to define an additional operation `merge(shift, K, L)` (where $0 \leq \text{shift} \leq D$). The operation returns the sequence of length $D + \text{shift}$ given by $k_1 k_2 \dots k_D l_{D-\text{shift}+1} \dots l_D$. In order to complete the routing of a particular message, L steps are required, where

$$L = D - \text{shift}. \quad (5.3)$$

Then, the shortest path is calculated by Algorithm 5.3.2. However, this algorithm does not consider the enhanced de Bruijn digraph as shown in Figure 4.7.

Algorithm 5.3.2: `SHORTEST_PATH(K, L)`

```

shift = 0
while (shift_match(shift, K, L) == false and shift < D)
  do shift = shift + 1
return (merge(shift, K, L))

```

In order to develop an algorithm for the enhanced digraph it is necessary to define the following operations on the de Bruijn based GUIDs: (i) `similarSymbol(K)` and (ii) `inverseSimilarSymbol(K)`, where K is a de Bruijn based GUID. The first operation returns the k_D symbol of K if and only if $k_i = k_D, \forall i \in [(D/2) - 1, D]$, otherwise a symbol that does not belong to the alphabet of the de Bruijn digraph. Similarly,

the operation `inverseSimilarSymbol(K)` returns the k_0 symbol of K if and only if $k_i = k_0, \forall i \in [0, (D/2) + 1]$, otherwise a symbol that does not belong to the alphabet of the de Bruijn digraph. Algorithm 5.3.3 provides the `enhanced_next_hop` algorithm for enhanced de Bruijn digraphs. The variable `routingTable[s]` represents the local routing table structure, which provides the address of the next hop that matches the provided symbol s . The entries of `routingTable` are populated based on the received ClusterMaps of the neighbor clusters. It should be noted that the ' \emptyset ' symbol represents a character not included in the alphabet of the de Bruijn digraph.

```

Algorithm 5.3.3: ENHANCED_NEXT_HOP( $K, L$ )

 $shift = 0$ 
while ( $shift\_match(shift, K, L) == \text{false}$  and  $shift < D$ )
  do  $shift = shift + 1$ 
if ( $shift == 0$ )
  then
    return ( $NULL$ )
if ( $K.similarSymbol() \neq \emptyset$  and
     $L.inverseSimilarSymbol() \neq \emptyset$  and
     $K.similarSymbol() \neq L.inverseSimilarSymbol()$  and
     $shift > D/2$ )
  then
    return ( $routingTable[K.similarSymbol()]$ )
  else
     $s = L[D - shift]$ 
    return ( $routingTable[s]$ )

```

Algorithm 5.3.3 is called on each Router to determine the peer of the next cluster that is closer to the final destination.

5.3.2. Messages

This section provides a description of the messages that constitute the routing protocol. Table 5.5 describes the structure of the `LookupRequest` message. It includes a *key* representing the GUID of the queried item. `LookupRequest` messages are forwarded towards the cluster whose GUID matches best with the included key.

Table 5.5.: `LookupRequest` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Key</i>	Includes the GUID describing the queried item.

Table 5.6 describes the structure of the `LookupReply` message. It includes a *key* representing GUID of the queried item and the related indexing information (*IndexValue*). It is common technique to set the indexing information to merely the address of the owner of

the related service or resource. However, the indexing information may include a long list of addresses for very popular items. Efficient mechanisms can be introduced in deployed systems to handle such scenarios. **LookupReply** messages are delivered directly to the originators of the queries.

Table 5.6.: **LookupReply** message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Key</i>	Includes the GUID describing the queried item.
<i>IndexValue</i>	Includes the GUID of the indexing information that matches the key.

Table 5.7 describes the structure of the **PublishRequest** message. It includes a *key* representing the GUID of the queried item and the *OwnerAddress* of the advertised service or resource that matches the provided key. The **PublishRequest** message is delivered to the cluster that matches best with the included key and subsequently, the Indexers update their local structures with the newly advertised item. Typically, advertisements have an expiration timeout. It is the responsibility of the owner to refresh the published information.

Table 5.7.: **PublishRequest** message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Key</i>	Includes the GUID describing the published item.
<i>OwnerAddress</i>	Includes address of the owner of the advertised service or resource that matches the provided key.

After successfully advertising an item, the responsible Indexer provides a confirmation to the originator of the advertisement using a **PublishConfirmation** message. Its structure is described in Table 5.8.

Table 5.8.: **PublishConfirmation** message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Key</i>	Includes the GUID describing the published item.
<i>Confirmation</i>	Confirms the successful advertisement of the published resource.

5.4. Maintainer

5.4.1. Functionality

The Maintainer is the most crucial role of the Omicron architecture. As it has been described in Chapter 4, it has the highest reliability requirements. Peers assigned with

the Maintainer role constitute the backbone of the architecture. Their basic objectives are to maintain the targeted de Bruijn structure between neighbor clusters and to orchestrate the balanced organization of their cluster itself.

Depending on the observed peer lifetime, multiple Maintainers may be necessary to ensure continuous maintenance of the topology. Maintainers are responsible for providing updated ClusterMaps to the peers of the cluster and to the Maintainers of the neighbor clusters. Requests for participation by new members are also handled by Maintainers. As it has been described in Section 4.5.1, peers perform random walks to collect samples on the endurance of the clusters and direct new peers to the clusters that have the greatest need for them, resulting in a balanced network infrastructure. The performance of this mechanism is evaluated with simulation experiments in Chapter 8.

Moreover, when the cluster population and subsequently the maintenance cost increase considerably, the cluster is divided in d new clusters (where d is the degree of the de Bruijn nodes), as long as the endurance requirements are met. Similarly, Maintainers trigger the cluster merging operation when their population is crucially low. The merging operation may be avoided if it is possible to migrate peers from neighbor clusters (without decreasing their endurance to a value less than a predefined threshold).

5.4.2. Messages

This section provides a description of the messages that constitute the routing protocol. Table 5.9 describes the structure of the `SplitClusterRequest` message. This message is exchanged among the Maintainers of the same cluster to initiate the division of the cluster (as long as the endurance requirements are met) into d new ones, where d is the cluster degree (recall that cluster represent nodes in the constructed de Bruijn graph). The `SplitClusterRequest` message includes a field with a list of the proposed new *ClusterMaps*.

Table 5.9.: `SplitClusterRequest` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>ClusterMaps</i>	Provides a list of suggested ClusterMaps by dividing the existing cluster members.

Maintainers that receive a `SplitClusterRequest` message check the validity of the suggested division and confirm it using a `SplitClusterConfirmation` message. The structure of this message is described in Table 5.10.

Table 5.10.: `SplitClusterConfirmation` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Confirmation</i>	Confirms the suggested cluster division.

In certain scenarios it is necessary to merge existing clusters that do not fulfill the endurance requirements. In this case a Maintainer of the critical cluster sends a merging request to a neighbor cluster using a `MergeClusterRequest` message. The structure of this message is provided in Table 5.11.

Table 5.11.: `MergeClusterRequest` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>ClusterMap</i>	Provides the ClusterMap to be merged.

Maintainers that receive a `MergeClusterRequest` message process the included ClusterMap. If the particular cluster represents the best candidate among the neighbors it accepts the merging request and it confirms it using a `MergeClusterConfirmation` message. The structure of the confirmation message is provided in Table 5.12.

Table 5.12.: `MergeClusterConfirmation` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Confirmation</i>	Confirms the suggested cluster merging operation.

5.5. Indexer

5.5.1. Functionality

Indexers are important roles that maintain efficient structures of the advertised resources and services. Their basic purposes are (i) to reply to incoming queries based on the indexed information, (ii) to store new resource and service advertisements and (iii) to synchronize the content of the local indexing structures with the content maintained by neighbor Indexers of the same cluster.

The indexing mechanism can be as simple as a hash table where GUIDs are used as keys and owner peer addresses as values. However, more advanced systems may need more complex structures such as rich metadata to provide advanced functionality, e.g., for range queries. The investigation of such mechanisms is out of scope for this thesis.

Moreover, depending on the type of the stored information, consistency can become a crucial requirement. In scenarios where a small number of Indexers suffices to provide the content, traditional consistency mechanisms can be efficiently applied [TN97].

5.5.2. Messages

This section provides a description of the messages that constitute the routing protocol. Periodically or after a certain number of indexing structure updates, Indexers syn-

chronize their locally stored indexing information to achieve a certain level of consistency. `UpdateIndexRequest` messages are used for this purpose. The structure of an `UpdateIndexRequest` message is given in Table 5.13. The included *index* can be the complete indexing structure. Alternatively, peers may select to exchange only recent differences in order to reduce the generated traffic load.

Table 5.13.: `UpdateIndexRequest` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Index</i>	Provides an updated index of objects for which the relevant cluster is responsible.

Upon the reception of an `UpdateIndexRequest` message, Indexers check the validity of the information. If they accept the incoming information they provide a confirmation using an `UpdateIndexConfirmation` message. Otherwise, they reply using a valid `UpdateIndexRequest` message back to the originator of the received message. The structure of an `UpdateIndexConfirmation` message is provided in Table 5.14.

Table 5.14.: `UpdateIndexConfirmation` message structure.

Field	Description
<i>Common</i>	Derived common message structure (cf. Table 5.1).
<i>Confirmation</i>	Confirms the valid update of the local Index.

5.6. Cacher

5.6.1. Motivation

Structured overlay networks for Peer-to-Peer (P2P) systems, e.g., Chord [SMLN⁺03], Pastry [RD01a], Tapestry [ZHS⁺04] and Omicron [DMS04], use proactive mechanisms to provide efficient indexing functionality for advertised resources. The majority of their implementations provide theoretical upper bounds on the communication cost in worst case scenarios. Nevertheless, these bounds assume that the topology maintenance process heals the divergence (caused by the dynamic participation of the peers) from the "ideal" network structure, e.g., a hypercube or a butterfly topology. By modeling the topology of a P2P network with a graph, the maximum distance between any two nodes is equal to the *diameter* of the graph. In graphs representing networks such as Chord each node maintains $O(\log(N))$ neighbors, where N is the number of nodes. The diameter of these networks is $D_{CH} = O(\log(N))$. The number of nodes may be equal to the population of the peers (e.g., in the cases of Pastry or Chord) or equal to the number of the constructed clusters of peers. An example of the latter case is the two-tier architecture of Omicron $D_O = O(\log(N/K))$, where K is the average population of each cluster.

Graph diameter defines the worst case shortest path between two peers. A more useful metric to evaluate the communication cost for routing messages in structured overlay networks is the *average inter-peer distance*. On the one hand, the average cost for graphs such as the one representing Chord is $\mu_{D_{CH}} = D_{CH}/2$ [SMLN⁺03]. On the other hand, the average inter-peer distance for networks such as Omicron based on de Bruijn graphs [dB46] is $\mu_{D_O} \simeq D_O - (k - 1)^{-1}$, where k is the degree of the nodes [LKR03]. However, since the graph nodes in Omicron represent clusters of peers, the actual average inter-peer distance is smaller than the average inter-peer distance in Chord.

Structured overlay networks have been designed mainly to overcome the intrinsic scalability issue of *flat* and *unstructured* networks, such as Gnutella v0.4 [PSAS01]. However, for several reasons, structured overlay networks have not been utilized in widely-deployed P2P systems (with the exception of the Kademlia network [MM02]). Instead, system designers opt for *hierarchical* or *hybrid* approaches where a subset of peers (usually termed as *super-peers*, or *ultra-peers*) is responsible for indexing and finding the advertised resources. Moreover, a number of mechanisms have been suggested to improve the performance of unstructured networks, e.g., expanding rings or multiple random walks [LRS02], [YGM02]. The success of these mechanisms is based on the assumption of uneven popularity of the available resources. In fact, this assumption is validated by a number of empirical observations of file sharing systems (cf. [SGG02], [GDS⁺03] and [BBR04]) where the popularity of the resources is reported. While there is a disagreement on the exact distribution that describes the popularity of the resources (i.e., Zipf or lognormal), it can be safely concluded that it is not uniform.

Therefore, an interesting debate has arisen lately on whether structured overlay networks can perform efficiently if non-uniform popularity of resources is observed [LCP⁺04]. Apparently, structured networks perform equally well for any lookup request, thus, providing upper bounds, though not exploiting effectively the query frequency. Some hybrid approaches have been suggested to address this issue, such as hybrid PIER [LHSH04] or OceanStore [RK02]. Though, in these hybrid approaches the formation of two separate overlay networks is suggested. On the one hand, the first overlay network is structured aiming to deal with unpopular queries. On the other hand, the second overlay network is unstructured targeting to deal with popular queries. The shortcomings and weaknesses of these solutions are mainly (i) the increased complexity, (ii) the additional maintenance cost that requires out-of-band signaling, (iii) the lack of adaptability to both uniform and non-uniform distributions and (iv) the increased delay when the initial overlay network selection for searching the resource fails and the fall-back alternative must be followed.

The aforementioned concerns are taken into account in the solution investigated in this thesis. A simple though efficient mechanism is suggested that capitalizes on the adequateness of caching resources following non-uniform distributions and the higher interest of the P2P users to a relatively small subset of the available resources. It extends the capabilities of structured overlay networks without any additional maintenance effort and very low additional routing cost compared to the original algorithms for structured networks in worst case scenarios where the cache is not properly updated. No extension of their signalling protocols is required. Thus, it avoids to further increase the complexity of

their operation³⁰. Merely, we invest on existing information collected through the normal network operation to improve the routing performance. The observed churn rate of P2P networks is the most critical factor (together with the popularity distribution) for the successful application of a caching mechanism. Both factors are considered in the performed simulation experiments provided in Chapter 8. While caching methods have been proposed for unstructured or hybrid overlay networks (cf. [Mar02], [LXN04]), these methods lack investigation on the structured counterparts. Moreover, several caching mechanisms have been extensively used for increasing the performance of Web technologies [AFJ00] and some of them use the P2P paradigm [IRD02]. Also, caching mechanisms have been utilized for content replication in P2P communities [Kan02].

5.6.2. Index Caching Mechanism Design

The rationale behind the caching mechanism is described as follows. Since peers participate both in generating queries and routing them towards the destination, it may be advantageous to reuse the information gained from the replies they receive from locally generated queries. Thus, peers may directly provide the position of the requested resource instead of forwarding the query until it reaches the final DHT destination. Moreover, if peers monitor the popularity of forwarded requests, they could additionally consider caching the most popular of them, provided that they hold the necessary indexing information. A simple mechanism to develop such indexing knowledge is to modify the semantics of the routing procedure. For popular requests, intermediate peers may consider to store locally the incoming queries and generate identical ones (though originated at the intermediate peer) and forward them instead of the original queries. The received replies can be used both to reply to the stored pending queries and to populate the local cache with useful and popular information. However, the gathered information may be used for a maximum amount of time t_{Th} that depends on the peer uptime distribution [DMS05].

The proposed scheme is illustrated in Figure 5.2 using a Chord-like structured network. There, at time t_1 peer Q_1 queries for a resource indexed at peer D (Figure 5.2(a)). Assume that peer I considers that the specific query is popular. Then, instead of forwarding the query, peer I generates an identical query that eventually arrives at peer D . Peer D replies to peer I , which both updates the local cache and provides the reply to peer Q_1 . Apparently, peer Q_1 may also update its local cache if it considers the query popular. Afterwards, assume that at time t_2 , with $t_1 < t_2 < t_1 + t_{Th}$ (where t_{Th} is the threshold time indicating that the cache content is valid with high probability), peer Q_2 queries for the same item and peer I is in the path towards peer D . In that case peer I provides the cached information to peer Q_2 immediately, skipping the rest of the lookup steps towards D (Figure 5.2(b)). Furthermore, peer Q_2 may update its local cache if it considers the query popular. However, in the latter case it is important to consider the "aging" of the information as it is not directly provided by peer "D", but from a cached index. Peer Q_2 has to set the lifetime of the entry in the cache to $T'_{Th} = T_{Th} - (t_2 - t_1)$ so that it considers the elapsed time since the advertisement of the item.

Two important factors drive the design mechanisms of caching. First, the scalability of the

³⁰Usually structured networks have more complex operation than their unstructured counterparts.



The second critical factor that has to be considered is the high churn rate of the peers. Nonetheless, conditional reliability mechanisms [DMS05] may reduce the side-effects, which are deeply investigated in Chapter 6. Naturally, popular resources are being held by several peers. Assuming that the responsible DHT nodes can provide back either the complete set of these peers or an adequate subset of them, the intermediate peers have sufficient information for locating a reliable peer that is still alive.

Figure 5.3.: Abstract description of the cache structure.

81

stores this time. The fifth field includes the list of collected *Indices* about peers that hold the requested resources and may be directly contacted. The subsequent field contains the list of *Pending queries* for this resource. Finally, the *Marked* field indicates that the cache replacement algorithm has selected this entry to be removed from the cache. However, the list of pending queries for this resource is not empty and the deletion of the selected entry has to be delayed until the reply will be received and the pending queries replied.

Further, an additional characteristic that may successfully be exploited to increase the efficiency of the structured networks is the fact that peers are also owners of resources. In cases where the requested resource is being held locally on the intermediate peer, it can safely be provided to the requestor. It may additionally be argued that instead of developing the index caching mechanism, intermediate peers can provide the requested resources themselves. Nevertheless, this possibility is application depended and many factors have to be considered, e.g., copyrights, technical limitations and system design. Moreover, if further constraints apply, e.g. find a resource or service provider in the closest vicinity to the requestor, this solution may not provide optimal performance.

5.6.3. Algorithms

Several cache replacement policies have been developed to meet the requirements of different problems (cf. least frequently used (LFU) [RD90], least recently used (LRU) and LRU-K [OOW93]). In fact, the replacement policy adopted for the index cache on each peer is a variation of the LFU algorithm, which is further enhanced with timeouts on the maximum lifetime of each entry. The latter improvement is mandatory for capturing the dynamics of P2P overlay networks. The pseudo-code of the LFU variation is provided in Algorithm 5.6.1. If there is an entry with 0 popularity and no pending queries, then this entry is removed. Otherwise, the least popular entry is returned³¹.

Algorithm 5.6.1: LFU_REPLACEMENT(*cache*, *pendingQueries*)

```

found = cache.get(1)
for i ← 2 to cache.size()
    {
        queryList = pendingQueries.remove(i)
        if (cache.get(i).popularity == 0 and queryList.isEmpty())
            then { cache.remove(cache.get(i))
                  return (null)
            }
        else if (found.popularity > cache.get(i).popularity and
                 (not cache.get(i).isMarked()))
            then { found = cache.get(i)
            }
    }
return (found)

```

The pseudo-code for filling a cache entry with information obtained from a reply is listed in Algorithm 5.6.2. Upon the reception of the reply all the pending queries are further

³¹The popularity of an entry on a particular peer is calculated by the number of related queries traversing this peer over the last time window.

replied. Moreover, if the cache entry is not marked, it is filled with the received index information.

Algorithm 5.6.2: `FILLCACHEENTRY(cache, entry, pendingQueries)`

```

queryList = pendingQueries.remove(entry.getEntryID())
for i ← 1 to queryList.size()
    do {
        lookupMsg = queryList.remove(1)
        lookupMsg.setDestination(lookupMsg.getInitiator())
        lookupMsg.setSender(localGUID)
        lookupMsg.setValue(entry.getValue())
        replyMessage(lookupMsg)
    }
if (entry.isMarked())
    then {myCache.remove(entry)}
    else {entry.setValue(srcs)

```

Finally, the pseudo-code for retrieving a stored entry from the cache is listed in Algorithm 5.7.1. If the stored entry is older than a safety time threshold (that is set based on the expected peer uptime) the entry is removed and a new one is created. Otherwise the frequency field is updated. Further, if the frequency of the query is higher than a threshold then, the message is stored as a pending query and a new lookup message gets created for the queried GUID, if this is the first pending message³². Moreover, if the size of the cache has exceeded its maximum value, the least frequently used entry is either removed if no pending queries are present or is marked for deletion at the arrival of the reply.

5.7. Inter-role Scenarios

In this section, some basic scenarios involving multiple peer roles are described in order to make the inter-role relationships and the way they co-operate to provide the targeted operations and services more comprehensive. In particular, three scenarios describing the necessary steps to complete a lookup request, a resource advertisement request and peer join request are discussed.

5.7.1. Lookup Request

In this scenario, the Router role is the main entity orchestrating the required steps. Figure 5.4 shows the involved actions³³. The first step is triggered by an incoming `LookupRequest` message (*step 1*). The local message dispatcher directs this message to

³²In this case, it should be noted that the returned entry contains no indexing information to indicate the status of the query to the routing mechanism.

³³In the illustrated scenarios, cascaded oval shapes are used to indicate similar roles belonging to different (neighbor) peers. Entities can communicate either by using message exchanging or by local method calls.

the Router, which checks whether the local cluster is responsible (*step 2*). The cluster identifier and the requested key included in the **LookupRequest** message are compared to make this decision. If the comparison result is positive then the request may be forwarded to a neighbor peer in the same cluster that has been assigned with the Indexer role (*step 2A*), unless the Indexer role is assigned to the local peer, too.

Algorithm 5.7.1: GETCACHEENTRY(*cache, id, pendingQueries, msg*)

```

entry = cache.get(id)
if (entry == null)
    then {
        entry = createNewCacheEntry(id, null)
        cache.put(id, entry)
        return (entry)
    }
if (entry.hasExpired() and ( not entry.isMarked()))
    then {
        cache.remove(id)
        entry = createNewCacheEntry(id, null)
        cache.put(id, entry)
    }
    else {entry.updateUsage()}
if (entry.frequency > FREQUENCY_THRESHOLD and
    entry.getValue == null)
    then {
        queryList = pendingQueries.get(id)
        if (queryList.isEmpty())
            then {
                lookupMsg = createLookupMessage(id)
                forwardMessage(lookupMsg)
                queryList.add(msg)
            }
    }
if (cache.size() - marked >= MAX_CACHE_SIZE)
    then {
        removed = LFU_Replacement(cache)
        queryList = pendingQueries.remove(removed.getEntryID())
        if (queryList.isEmpty())
            then {myCache.remove(removed)}
        else {removed.mark()}
    }
return (entry)

```

Afterwards, there are three options on how to proceed. The first alternative is followed if the Indexer role has been indeed assigned to the local peer. Therefore, the Indexer role is called and asked to checked whether the requested object has been published and is currently available (*step 3A*). Then, the reply is provided back to the Router (*step 4A*). The second option is followed if the Cacher role has been assigned to the local peer. Thereby, the Router checks whether the requested object is cached in the local cache (*step 3B*). Then, the reply is provided back to the Router (*step 4B*). The third option is followed either if the Cacher role is not assigned to the local peer or if the requested object is not in the local cache. In that case, the request is forwarded to the next peer that is closer to the targeted cluster (*step 5A*). If the requested item is included either in the local index or cache, the requestor peer is contacted and supplied with the requested information (*step 5B*), though.

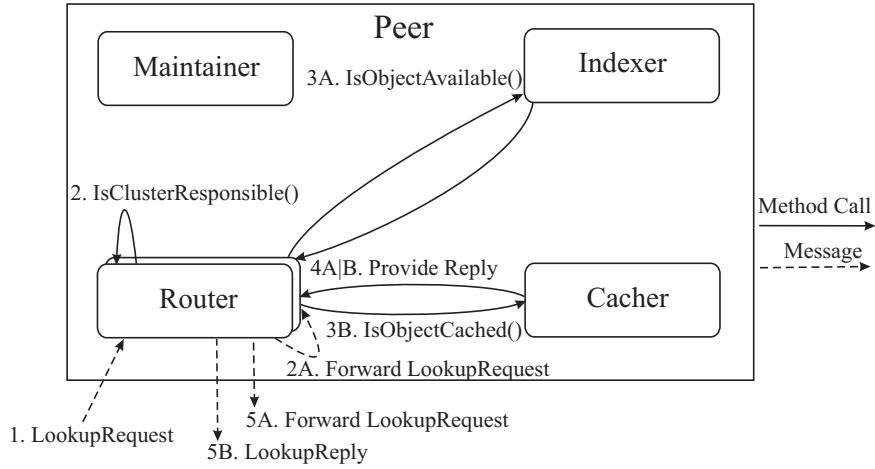


Figure 5.4.: Lookup scenario steps.

5.7.2. Resource Advertisement

The second scenario, which is graphically shown in Figure 5.5 involves the Router and the Indexer. This scenario is triggered by the reception of a **PublishRequest** message (*step 1*). Similar to the previous scenario, the message is directed to the local Router, which checks whether the local cluster is responsible (*step 2*). The cluster identifier and the generated key for the advertised object included in the **PublishRequest** message are compared to make this decision. If the comparison result is positive then the request may be forwarded to a neighbor peer in the same cluster that has been assigned with the Indexer role (*step 2A*), unless the Indexer role is assigned to the local peer as well. Then, the Indexer role is called and the local index is updated to include the newly advertised object (*step 3*). The local Indexer sends **UpdateIndexRequest** messages to neighbor Indexers belonging to the same cluster (*step 4A*). Based on the applied consistency policy, an additional action may periodically take place at this stage. Nevertheless, the Indexer confirms to the Router that the newly published object has successfully been advertised (*step 4*), which subsequently provides the confirmation back to the originator of the **PublishRequest** message (*step 5B*). Alternatively, if the local cluster is not the appropriate place to advertise the new object, then steps 3 and 4 are skipped and the Router directly forwards the request to the next peer that is closer to the targeted cluster (*step 5A*).

5.7.3. Join Request

The third scenario is intrinsically different from the previous scenarios. It is triggered by the reception of a **JoinRequest** message (*step 1*). Such a message can be handled only by peers assigned with the Maintainer role. Thus, as it has been described in Section 4.5.1, this request is always redirected by other peers to Maintainers of the same cluster. The scenario is graphically shown in Figure 5.6.

The Maintainer initiates a random walk randomly traversing selected neighbor Maintainers

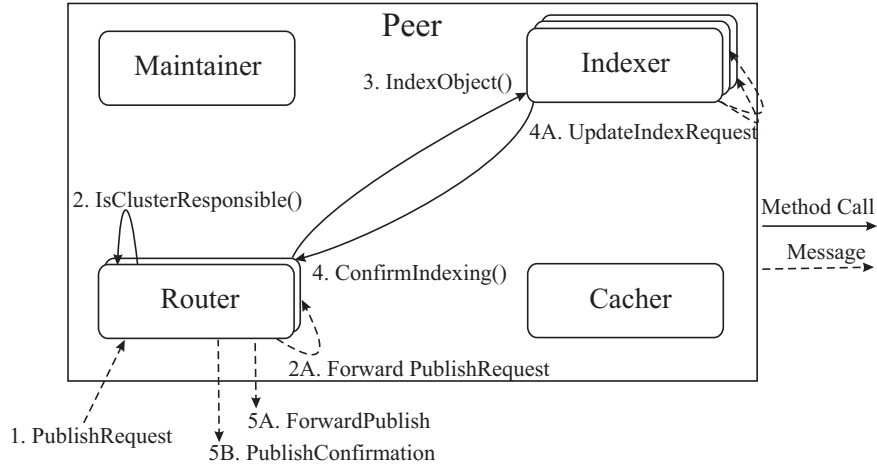


Figure 5.5.: Resource advertisement scenario steps.

that are members of neighbor clusters (*step 2*). After the appropriate cluster for accepting the new peer has been selected, the Maintainer must inform the joining peer. However, roles installed at a peer can communicate only with similar roles in remote peers. Since the new peer may not have been assigned with the Maintainer role, it is not possible for the Maintainer to accept the new peer. Thus, the *Connector* role has been introduced to fill this design gap. The Connector can provide three important pieces of functionality: (i) *connectivity functionality*, (ii) *new role installation functionality* and (iii) *ClusterMap updating functionality*.

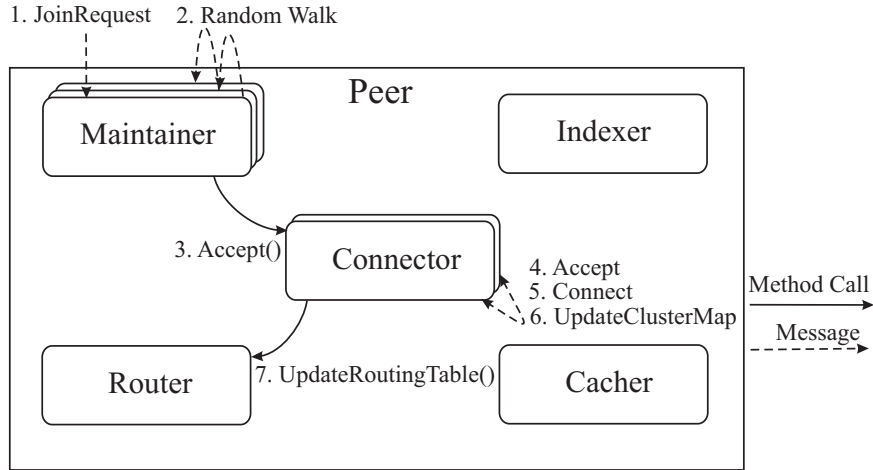


Figure 5.6.: Peer join scenario steps.

Therefore, the Maintainer calls the Connector to accept the new peer (*step 3*). Subsequently, the local Connector contacts the remote Connector and provides the acceptance decision (*step 4*). Then, the new peer requests from the Maintainer to be considered as a member of the same cluster (*step 5*). Afterwards, the Maintainer provides the local and the neighbor ClusterMaps (*step 6*) to the new peer, which can populate its routing table

with valid addresses (*step 7*).

5.8. Summary

The specialization based approach introduced with the role model provides a advantageous mechanism to exploit heterogeneous populations of peers. This chapter has provided the detailed description of the identified core roles by focusing on their functionality, the employed algorithms and mechanisms as well as the related protocols.

The common characteristics of the utilized messages have been identified in order to develop them based on a base structure and furthermore, to handle them with common mechanisms, i.e., message dispatcher. Moreover, an effective inter-role communication model has been designed to unambiguously define the protocols used by each role. An additional default role has been developed to provide the basic functionality required by each peer of the system.

Special attention has been paid to design an efficient routing mechanism. The original structure of de Bruijn graphs has been enhanced by removing the self-connected links of particular nodes. Thereby, the optimal routing algorithms found in literature have been adapted to take advantage of the new structure and to provide a better balance of routing traffic.

Moreover, the adequacy of caching popular indices in intermediate peers along the paths towards the responsible indexing peer(s) for structured networks is demonstrated in this chapter. The proposed caching mechanism considerably reduces the routing cost in structured P2P networks. Compared to alternative proposals, the achieved performance improvement is combined with a set of additional attractive features. Since the mechanism is locally applied to peers it can be deployed incrementally. Moreover, there is no need to introduce multiple specialized overlay networks operating in parallel or additional protocols to update the cached information.

Inter-role scenarios have been provided to investigate the way roles co-operate and support each other aiming at efficient system operation. Three particularly common scenarios provide the necessary description of the required steps to accomplish a lookup request, a resource advertisement request and peer join request.

Optimal Role Assignment

Do not trust all men, but trust men of worth; the former course is silly, the latter a mark of prudence.

DEMOCRITUS

Conceptual outline.

The peer role model of the Omicron architecture has been introduced in Chapter 4 and subsequently investigated in more detail in Chapter 5. Moreover, we have seen in Chapter 3 some alternative design approaches that assign distinguishable roles to different peers aiming to address their heterogeneity. Admittedly, the assignment of a role to a peer is a costly operation especially for demanding roles, i.e., Indexers and Maintainers. Therefore, achieving optimal system operation requires to assign the demanding roles to the most reliable peers. However, such a knowledge is not a priori available. Conditional reliability mechanisms (aka. burn-in mechanisms) can supply a technique to improve system's performance and maximize the stability of the network. A condition to apply this technique is related to peers' departure hazard rate. Thereby, it is mandatory to investigate the available empirical observations and model peers' continuous availability with a parametric distribution. Optimal role assignment is the challenging procedure of selecting the appropriate assignment time without having knowledge of each peer's reliability. We develop a cost-effective model for optimal role assignment and demonstrate its advantages to an example P2P system. Moreover, the observed results are applied in the estimation of clusters' endurance.

This chapter is organized as follows. The significance of this work is motivated in Section 6.1, followed by an overview of the related theoretical background in Section 6.2. In Section 6.3 the empirical details of interest found in related literature are analyzed and a basic matching of the observed data with single probability distributions is performed. In Section 6.4 a general peer life-cycle model is developed, which is later customized to the characteristics of file sharing services to obtain a meaningful parametric distribution that fits nicely the observation curve. Section 6.5 describes the optimization procedure and the achieved reliability performance. Using the obtained results, Section 6.6 demonstrates the estimation of the endurance of a cluster. Finally, Section 6.7 summarizes this chapter.

6.1. Motivation

A large number of measurement reports studying P2P systems have found that the majority of peers participate for a relatively short amount of time only [MM02]. Such behavior (which has been observed in Internet-wide file sharing applications) results in highly fluctuating and frequently unstable overlay network topologies.

Unstructured overlay networks, e.g., Gnutella v0.4, follow a reactive approach with respect to the actions taken when peers join. Indeed, in the reactive approach there is a very low cost at each peer arrival; however the communication cost increases significantly with the rate that peers generate queries [PSAS01]. A number of techniques have been investigated to decrease this communication cost and promising results have been reported especially when requesting popular items [LRS02]. Though, in worst case scenarios they still perform inferiorly compared to structured networks especially in scenarios of looking for unpopular items.

On the other hand, structured overlay networks based on *distributed hash tables* (DHTs), e.g., Chord [SMK⁺01] or Pastry [RD01a], follow a proactive approach. On each peer arrival a number of actions takes place, viz. to place the new peer at an appropriate position of the network topology, to update its neighbors' routing tables, to reassign the indexing responsibilities and to advertise the newly shared services and resources. In more advanced systems, additional actions might be performed as well. Structured overlays can provide much lower worst case upper bounds on lookup operations than their unstructured counterparts³⁴. However, they suffer considerably from frequent peer arrivals and departures and more specifically from the short durations some peers stay connected to the system. The term *peer uptime* or simply *uptime* is adopted in the rest of this thesis to denote the time a peer remains online after joining without interruption. Alternative terms found in the related literature are *peer on-time* and *peer lifespan*. A limited study of DHT-based structured networks is given in [LSG⁺04].

To elaborate further into the issue, consider a worse-case scenario where malicious users deliberately exploit the proactive approach and attack a P2P system designed as a structured overlay network by frequently joining and leaving. Such attack scenarios have been described in the P2P research community (cf. [CDG⁺02] and [SM02]). The potentially large amount of advertised information on each join can cause a hard to defend *distributed denial of service* (DDoS) attack. Moreover, based on the weaknesses of the replication policy applied, vital shared information such as indexing structures or even more important data (e.g., trust-related accounts) might be lost as responsibilities are assigned to malicious peers.

Apparently, hybrid overlay design approaches have the potential to handle more efficiently the aforementioned issues. For instance, the core JXTA virtual network [TAA⁺03], the KaZaA system [LRW03] or any other approach that incorporates the concept of *super-peers* can handle high churn peer rates more effectively. In the case of Omicron (that employs a rich and well balanced role model), it is suggested to assign demanding roles

³⁴In most DHT-based approaches it takes $O(\log(N))$ steps, where N is the active peer population [SMK⁺01], [RD01a], [DMS04].

that trigger costly actions to peers that have proved their stability, while still making a profitable usage of the unstable ones. More specifically, taking advantage of the statistical properties of the measured peer reliability, it is suggested to assign low cost roles to newly joined peers. As peers remain connected for a longer time the hazard rate of departing from the system is expected to be decreased. Costly roles are assigned afterwards to the statistically more reliable nodes.

However, it is still a challenge to determine the reliable peers as early as possible. As a matter of fact, similar problems have been addressed in other research domains, too. For example, in the procedure of manufacturing silicon-based integrated circuits (ICs), a high rate of the so-called *infant mortality* is exhibited. That is, ICs tend to show their defectiveness very early in their operational lifetime. A standard engineering method used to identify such defected components is called *burn-in* [KK83]. Using burn-in, engineers can test the newly manufactured components under certain environmental conditions and remove the defected components.

In this chapter, we investigate in depth the details of applying the burn-in method in the construction of efficient and reliable P2P overlay networks. A critical issue that arises is selecting the optimal time to assign a demanding role to a peer, so that both the risk of failure will be significantly decreased and the system will benefit maximally from the reliability of the stable peers. The investigation of this subject is the aim of this work. However, a condition to effectively apply the burn-in method is to have a decreasing failure rate in the peer lifetime. A thorough exploration of the related reports in the literature acknowledges this fact; nevertheless, the numerous suggested distributions capturing the peer uptime differ considerably. In this chapter, a representative set of empirical observations is decomposed and analyzed, thereby, following a different path for extracting the distribution characteristics than the common treatment applied in most analysis cases. Meaningful interpretations of the results are provided, taking into account the intrinsic characteristics of the underlying process that drives peers' uptime as well as practicality aspects such as the heterogeneity of the peers.

6.2. Distribution Generative Models

Before proceeding with the analysis of the empirical observations, it is worthwhile reviewing some general but interesting results from the related probability theory literature. In fact, there is a great debate in the more general Internet community on whether certain quantities follow lognormal or Pareto distributions, e.g., the size of Web files [Dow01b], [GLMT01]. As it will be more clear in the following section, opinions also differ on whether approximating the peer uptime with either a lognormal or a Pareto distribution to get better results.

A thorough investigation of the relation between lognormal and power law distributions is provided in [Mit03]. What is becoming clear from this work is that very similar *generative models* lead to either power law or lognormal distributions, depending on seemingly trivial variations.

For instance, it is well known that the lognormal distribution follows the law of proportionate effect and may be the result of many small multiplicative factors. Taking an example from biology, a way to describe the growth of an organism is by employing such *multiplicative processes*. Assuming that the starting organism has size X_0 , the size of the organism can grow or shrink on each step j of the process by multiplying the size of the previous step with a random variable F_j :

$$X_j = F_j \cdot X_{j-1} \quad (6.1)$$

Such a process results in a random growth of the size that is independent of the exact current size but is expressed as a percentage of it. Based on the Central Limit Theorem [Fel71, page 258], as long as the factors $L_k = \ln F_k, 1 \leq k \leq j$ are independent and identically distributed variables with finite mean and variance, it can be easily shown that X_j asymptotically approaches the *lognormal* distribution.

Alternatively, if one assumes that there is a minimum size for the organism, say m , and the size of the first range includes sizes between m and γm for some $\gamma > 1$, the second range comprises sizes between γm and $\gamma^2 m$ while the j th range contains sizes between $\gamma^{j-1} m$ and $\gamma^j m$, we obtain a power law generative model. In particular, assuming that over each time step the probability of moving from class i to class j depends only on the value $j - i$ (denoted as p_{ij}) and setting $\gamma = 2$, $p_{ij} = 2/3$ if $j = i + 1$, and $p_{ij} = 1/3$ if $j = i - 1$, then the equilibrium probability of being in class p_k is $1/2^k$. Therefore, the probability of the organism size being in a class greater than or equal to k is $1/2^{k-1}$ and the complementary cumulative distribution function (CCDF) in equilibrium is given by:

$$Pr\{X \geq x\} = m/x \quad (6.2)$$

where $x = 2^{k-1}m$, which is a *power law* distribution [Gab99].

The difference between the two aforementioned models is that the latter (also known as the Champernowne model) assumes a minimum organism size; thus, generating a power law distribution. Assuming a minimum value can be valid for certain applications of the model, e.g., the size of cities. In contrast, the organism size in the multiplicative model can become arbitrarily small and therefore generating a lognormal distribution.

Similarly, for certain sets of empirical observations, it is difficult to distinguish whether a lognormal or a Weibull distribution provides a better fit [Cai02].

6.3. Empirical Observations

6.3.1. Measurements Analysis

As it has been already mentioned in the introduction, a number of measurement reports (cf. [MM02], [MTG03], [SGG02] and [SW04a]) have investigated the user behavior of the "first-generation" of widely-deployed P2P file sharing systems. All of them agree that the large majority of the population remains online for a short period of time only. However, it is reported in these measurements that there is a subpopulation that tends to have a

significantly longer uptime compared to the majority. In overlay network design, one can capitalize on this stable subpopulation to attain more effective solutions.

More specifically, the following conclusion is reported in [MM02] : *"The longer a node has been up, the more likely it is to remain up another hour"*³⁵. Moreover, as it is stated in [JP03], the measurements gathered in [SGG02] suggest that the peer uptime distribution follows a Zipfian distribution while a similar report in [SW04a] suggests a lognormal distribution. A different study [CLL02] reports among others that peer uptime follows a log-quadratic distribution³⁶ and it is suggested to be approximated by two Zipfian distributions. Finally, Ripeanu [Rip01] discovered for the Gnutella v0.4 system that about 40% of peers leave the network within less than 4 hours, while only 25% are alive for more than 24 hours. Summarizing these early reports, it can be concluded that the peer uptime distribution has in general a tail heavier than it is in the case of Poisson distribution; though the exact parameters might depend on the specific observation period and the examined system.

Nevertheless, these measurements have been performed more than three years ago and may not accurately reflect the user behavior as it can be observed today (though they are still cited frequently). The most important factor that makes the quantitative (and in some cases qualitative) details of these results questionable is the evolvement of the type of the shared media files. While in the past users mostly exchanged compressed music files (the size of which is typically ranged between 3 and 10 MBs), today users exchange compressed video files in DivX format (which are typically around 700 MBs) or even MPEG2 encoded DVD format (which can be more than 4 GBs large)³⁷. There is a change in the magnitude of the size of the requested material that is not matched by a similar growth in the available network bandwidth. This naturally results in longer downloading times and despite the fact that downloads can be safely resumed, a change in the user behavior has been noticed.

For example, some newer measurements have been performed on the so-called *"second-generation"* P2P file sharing systems based on the Gnutella v0.6 network (cf. [SR04] and [BQ03]), the KaZaA network (cf. [GDS⁺03], [LRW03] and [LKR04]), the eDonkey network (cf. [MTG03] and [Tut04]) as well as the BitTorrent network (cf. [IUKB⁺04] and [PGES04]). For instance, Stutzbach and Rejaie [SR04] studied Gnutella v0.6. They suggest either a two-piece power law distribution³⁸ or a log-quadratic one³⁹ to fit the peer uptime. The same network has been studied by Bustamante and Qiao [BQ03]. Here the authors approximate the peer lifespan with a power-law distribution⁴⁰.

In [LRW03] the authors focus on the shared content found in the KaZaA network and more specifically the file size distribution, the popularity distribution and the dynamic

³⁵It should be noted that this report analyzes the measurements provided in [SGG02] by observing the Napster and the Gnutella v0.4 networks.

³⁶The suggested distribution is approximated by the curve $y = 10^{-0.16x^2 - 0.61x + 0.35}$.

³⁷Two excellent books on the characteristics of numerous media types may be found in [SN04b] and [SN04a].

³⁸For $x < 118$, $p \propto e^{-1.89684 \log x}$ and for $x \geq 118$, $p \propto e^{-1.18356 \log x}$.

³⁹The suggested distribution is approximated by the curve $y = e^{0.148933 (\log x)^2 - 0.125981 \log x}$.

⁴⁰The suggested distribution is approximated by the curve $y = 4338.8x^{-1.0607}$.

availability of the content. What is interesting to note in this work is the file size cumulative distribution (cf. Figure 5 in [LRW03]). The two-step curve clearly supports our arguments on the evolvement of the shared content from mostly music files to a mixture of music and video (or software) files. From a different study of KaZaA [LKR04] that focuses on the connectivity of peers, one obtains additional information about the average connection duration of the leaf-peers and super-peers. Additionally, the super-peers' lifetime has been monitored. Despite the fact that the provided measured distribution has a similar shape compared to [SGG02], the average lifetime of the KaZaA super-peers is much higher (149 mins \simeq 2.5 hours). Finally, in [GDS⁺03] it is investigated how users' behavior changes as they age (termed as "*attrition*") in the KaZaA network. It is reported that new users are more active than older ones, which eventually leave permanently from the system. As an alternative explanation, we suspect that a subpopulation might obtain a different globally unique identifier, i.e., in cases of installing newer versions of the software that cannot be distinguished easily from the permanently departed peers.

BitTorrent, which is apparently one of the most popular P2P content file sharing systems today, follows a different system design approach compared to usual P2P file sharing applications. It uses the Web for content searching and only when a user has identified an interesting search result it connects to the network. The Web approach combined with the tracker-based architecture of BitTorrent have an important influence on the resulting lifetime of the peers; peers that join the network have already found interesting files via indirect means (the Web) and trackers have in general a longer lifetime expectation than usual peers. Moreover, it should be noted that BitTorrent follows a policy of giving higher priority to older users and apply a tit-for-tat accounting mechanism to motivate users to have longer uptime. In [PGES04] it has been measured that half the trackers have an average uptime of 1.5 days. The measurements found in [IUKB⁺04] describe the sessions' lifetime for sharing a relatively large size (1.77 GB) Linux Redhat distribution. It is reported that the large majority of the peers downloaded the Linux distribution in a single session, having an average duration of 8.1 hours. Thus, it can be safely concluded that a large number of BitTorrent users are very stable. Similar behavior can be noticed in a significant number of users of the DirectConnect network [SW04a], though this particular report is not updated to reflect the additional stability that has been recently observed in some hubs of this system.

Additionally, researchers performed some measurements considering the *session duration* between peers. Though session duration cannot imply that peers' uptime follow a similar distribution, it can provide some lower bounds for peer uptime. More specifically, Schollmeier [Sch05] processed the provided data in [SGG02] and accurately approximated the measured data with a two-parameter Weibull distribution (with parameters $\beta = 0.7$, $\eta = 90.9091$). In addition, Schollmeier collected measurements observing the hierarchical network of Gnutella v0.6. The session duration of both the leaf and the super-peer nodes have been approximated also with a Weibull distribution, (with parameters $\beta = 0.72$, $\eta = 800$ and $\beta = 0.42$, $\eta = 300$, respectively). Furthermore, de Meer et al. [MTG03] investigated the duration of each inter-peer connection and they accurately modeled it using a mixture of two lognormal distributions, one for short lived connections (with mean 1.85 and variance 9.99) and one for long lived connections (with mean 363 and variance 129000). Finally, Tutschku [Tut04] distinguishes between signaling and content

delivery sessions in the eDonkey network. The duration of the latter sessions is suggested to be approximated with a lognormal distribution.

6.3.2. Critical Factors

Such a diversity in the results of the measurement reports implies that at present there is no general valid model describing the peer uptime. Concluding from the previous discussion, there are some important factors that have to be considered in building a model that can describe adequately the user behavior and the way it is related to peer uptime. Some of these factors are related to the performed measurements while others point to more general issues.

Most of the measurements agree that the uptime distribution has a heavier tail than the Poisson distribution. However, the exact distribution cannot be safely described and the significance of the tail cannot be accurately evaluated. With the evolvement of the content sharing systems a number of widely accepted related reports have partly become obsolete. In particular, as the size of shared content gets larger (without a similar change in the available access network bandwidth) peers tend to have a longer uptime. Furthermore, the employed measurement techniques (i.e., crawler-based methods) collect (both left and right) censored data since peers' joining time takes place before they are captured by the crawler and their departure occurs before such an event is observed. Moreover, the total observation time of each reported experiment binds the ability to adequately and accurately capture both tails of peers' uptime distribution. Further, most of the investigations followed a *cumulative distribution function* (CDF) based fitting method to select an appropriate matching distribution. While some of the results get good matches for the head of the empirical distributions, they do not fit the tail correctly. Finally, it should be noticed that the reported measurements have been performed on a single application type, i.e., file sharing applications. There is a lack of reliable measurements in different applications that could further augment achieving a general trustable model to adequately and accurately describe the uptime for every P2P system.

In addition, there are key factors of more general concern to be taken into account. Particularly, the flat rate charging policies (i.e., for DSL connections) offered lately by many Internet Service Providers (ISPs) demoted the former motivation of users to limit the usage of the leased network resources. Moreover, incentive-based methods in P2P systems can significantly influence the user behavior provided that accounting mechanisms for asynchronous service consumption and provision motivate users to behave having longer term policies [LDMS05]. In fact, such mechanisms have been already deployed in the latest versions of many aforementioned systems. The tit-for-tat mechanism of BitTorrent, the contribution factor of KaZaA or the hordes formation of eDonkey are some of the examples.

6.3.3. Empirical Fitting

Considering the distributions suggested by the research community, it is necessary to revisit the fitting phase to get some better insights on the validity of the different approaches.

Most researchers fitted the empirical data based on the (linearly-scaled) CDF function. As already mentioned, such methodology does not provide optimal fitting for the tail of the distribution and can lead to questionable results. Alternatively, log-log CCDF graphs can provide a more accurate graphical-based estimation of the fitting in both the head and the tail of the distribution [Dow01a].

We performed a fitting of the empirical observation reported by Bustamante and Qiao [BQ03] who provide their data freely on the Web⁴¹. It should be noted that the observed data include peers uptime that range in the interval (1300 seconds, 3.5 days) due to unavoidable practical limitations of the crawler used to collect the data. Moreover, the authors utilized the "*creation-based method*" [RLA00] to improve the validity of the results at least for the interval of interest. Using this pessimistic method, only the entries that begin in the first half of the trace are accounted. We note this point in our diagrams with a vertical line to mark the validity area in the observations.

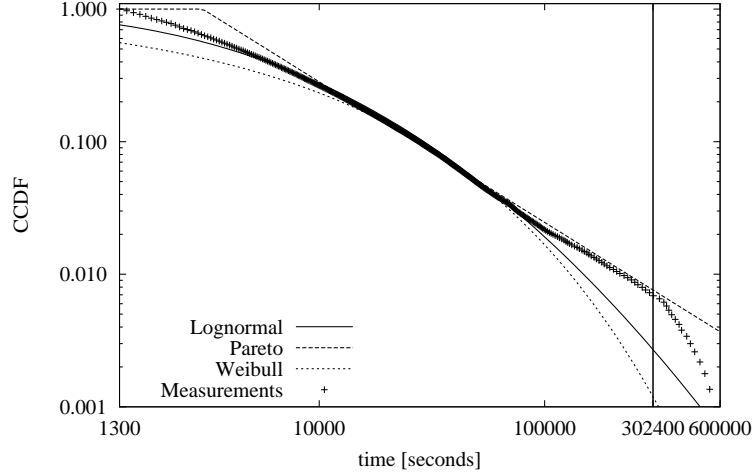


Figure 6.1.: CCDF-based fitting with single distributions of the empirical curve.

The criterion for the fitting process is to match the observations as close as possible to the "*waist*", which results in some error both in the head and the tail. The results of the fitting are shown in Figure 6.1. It displays the empirical curve matched with a Weibull, a lognormal and a Pareto distribution in a log-log CCDF graph. As it can be seen from this figure, the Weibull distribution underestimates the survival function at the head of the curve and has a much steeper tail. The Pareto distribution (which is suggested by the authors) does not match well the head, but it fits nicely the waist and the tail. Finally, the lognormal distribution fits well the waist, though it has an acceptable error in the head and a steeper tail. The parameters of the fitted distributions are provided in Table 6.1⁴².

Driven by the need to have an alternative view on the matching results, we have inves-

⁴¹The respective URL is: "<http://www.aqualab.cs.northwestern.edu/download/lifeTrace.tar.gz>".

⁴²The α parameter of the Pareto distribution is identical to the one suggested at [BQ03] by adding the offset time period of 1300 seconds.

Table 6.1.: Parameters of fitted distributions.

Distribution	Parameter 1	Parameter 2	Density probability function
Weibull	$\lambda = 0.0233$	$\eta = 0.449$	$f_{Wbl}(x) = \lambda \eta x^{\eta-1} e^{-\lambda x^\eta}$
Pareto	$\alpha = 3038.8$	$\beta = 1.0607$	$f_{Prt}(x) = \beta x^{-1} \left(\frac{\alpha}{x}\right)^\beta$
Lognormal	$\mu = 8.28$	$\sigma = 1.56$	$f_{Lgn}(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(\log(x)-\mu)^2}{2\sigma^2}}$

tigated the fitting of the aforementioned distribution in a log-log CDF graph where the focus is placed on the interesting part of the head. The results of the matching are shown in Figure 6.2 where the vertical axis is shifted to the right to avoid misinterpretations of the early part due to the lack of data (which would probably be underestimated due to practical limitations). The lognormal distribution provides the best approximation, while the Weibull is also close. The Pareto distribution is certainly inappropriate. Thus, it can be stated that the lognormal distribution provides the closest fit for the head while the Pareto matches the tail well. So far only simple distributions have been considered. It seems that there is enough room for an even better approximation, which will be further investigated in the following section.

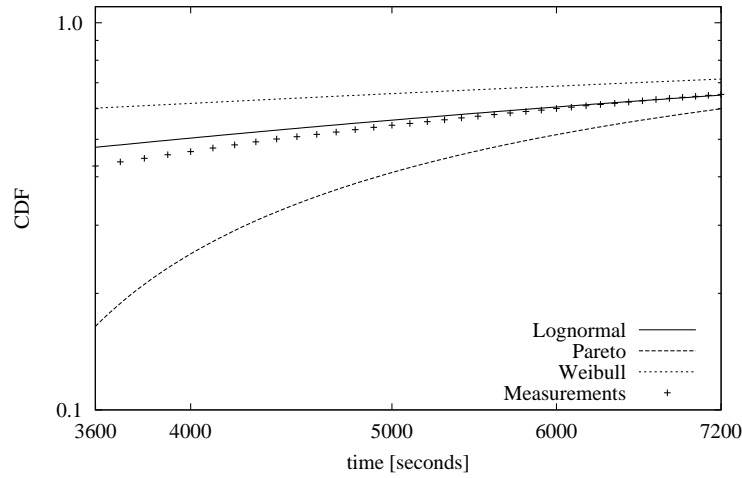


Figure 6.2.: CDF-based fitting with single distributions (head) of the empirical curve.

6.4. Modeling

6.4.1. Peer Life-cycle Model

Based on the previous discussion on the collected measurements and the concluded factors, a model of the user behavior focusing on the uptime distribution is being derived in this section. The model is expressed in as general terms as possible. However, certain

parameters are influenced both quantitatively and qualitatively by file sharing application related observations.

Typically, a user is motivated to join a P2P network by the wish to use certain services or resources. The user might have a general description of the required service or resource. Therefore, the first action performed is to search for an available service or resource that fits best with the initial description. Assuming that such a service exists and the user is satisfied, there might be an optional negotiation phase (especially in cases where advanced business models apply). Thereafter, the service consumption phase starts where the duration is context-depended. Additionally, the user might in parallel request further services from the system. The user is expected to remain connected until the services consumption is completed, motivated by the fact that the requested services are currently available, unless an external event interrupts the ongoing sessions. Such a scenario holds especially when there is a lack of availability guarantees and users adopt an opportunistic behavior. Of course, satisfaction of specific *quality of service* (QoS) requirements might be critical for the node uptime, e.g., above a certain transmission bandwidth for file downloading.

However, there is a probability of proceeding with unsuccessful steps in the aforementioned scenario. For example, the requested service might not be found, the negotiation phase might fail, the providing peer might depart before the service completion, the QoS might drop below the required threshold, etc. In this case, the probability that the user will depart earlier than expected is getting much higher. Though, replicated services is a common feature of most P2P systems, however, in many cases the demand is also high, thus, using up a potential surplus of available resources. Figure 6.3 provides a high level graphical description of peer's life-cycle capturing the basic states and their relationships. It is inevitable that peers will leave the network at some time. This fact is reflected in the transient nature of the top states and the absorbing nature of the bottom state. This progressive model will eventually develop through the transient path to the absorbing state with probability 1, although there might be backward movements and some reversibility; there is still a clear direction in the development.

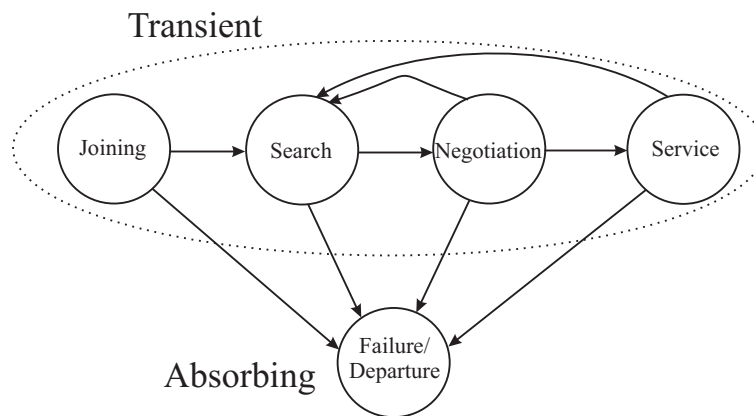


Figure 6.3.: Peer life-cycle model.

Despite the fact that such a high level description cannot provide a detailed mathematical model of a peer's life-cycle (mostly because it cannot tractably model the potential

"dynamics" as the process makes cycles in the transient states) it is an essential step to construct more formal descriptions, e.g., based on Stochastic Petri Nets (SPNs) [JMT94]. However, a detailed SPN-based model might have been very complex since it has to consider many factors and numerous application depended issues have to be taken into account. In fact, peer uptime is the time until the departure event occurs. This event is the result of an underlying *process*, which is in most cases largely unknown. In certain realizations where application-specific events (i.e., initiation of a new file download) could be captured and reported to the underlying overlay infrastructure, the structure of the underlying process can become clearer and augment its modeling. However, such a realization is not always feasible and certainly not a straightforward procedure.

6.4.2. Swarm-based File Sharing Model

In order to derive a more concrete model to describe peers' uptime, a specific application has to be considered. In fact, we have to consider a service that matches the characteristics of the one used to collect the available empirical data, thus, a file sharing application. Advanced file sharing services employ the so-called "*swarm*" technique to increase their performance. Using this technique, file sharing systems divide shared files into a number of blocks, usually of constant size. As a result, peers can download in parallel different blocks from different remote peers and increase the service throughput; every peer acts as a server supplying multiple other receivers. However, bandwidth limitations (usually forced by local policies or the access network) create the need for a queue on each peer where requests for blocks are stored and wait to be serviced based on the implemented scheduler that may be priority based.

An individual peer that joins a P2P file sharing network initiates a number of requests (in parallel or in series) and waits to be serviced by other peers. The time it takes to download the requested files is the result of a multiplicative process composed by mostly independent events for each block (where dependencies might occur when multiple blocks are served by the same remote peer). The achieved download bandwidth is a complex quantity to be computed analytically. It is out of scope of this thesis. Nevertheless, intuitively it can be stated that as this multiplicative process is progressing, the achieved throughput is the metric that mostly determines the lifetime of the peer (unexpected interrupts or faults can be another random source that terminates the process). Impatient or unsatisfied users interrupt peer's participation in the system. Furthermore, as there exist multiple file sharing systems, users try in parallel to find the requested resources and they remain connected only to the best performing system. Recalling our discussion in Section 6.2 and based on the fact that no lower failure time can be defined since users can depart at any arbitrary moment, a lognormal distribution can naturally be selected to describe peers' uptime.

However, the investigation in Section 6.3 concluded that a pure lognormal distribution does not match perfectly the empirical observations. Therefore, there is a need to consider further factors that drive the underlying process.

A very important aspect not considered so far is the *heterogeneity* of peers. Therefore, we have developed a mixture process of two lognormal distributions (since the lognormal dis-

tribution fits well with the multiplicative underlying process) that can augment capturing the heterogeneity. Two classes of peers have been considered, a stable one that represents peers connected to high bandwidth connections (e.g., T3) and a less stable one (e.g., using DSL connections). The ratio between those classes has been considered to be 20% stable and 80% unstable peers to reflect reality adequately. Having these constraints the shape of the mixture curve has been optimally fitted to that of the empirical one. Table 6.2 provides the estimated parameters for the two component distributions, namely *Lognormal A* and *Lognormal B*. The former distribution represents the unstable peers and is called the *weak* component, while the latter distribution represents the stable peers and is called the *strong* component. The resulting matching is graphically shown in Figure 6.4 where the almost perfect matching can be observed. Additionally, we have drawn the suggested Pareto distribution for direct comparison.

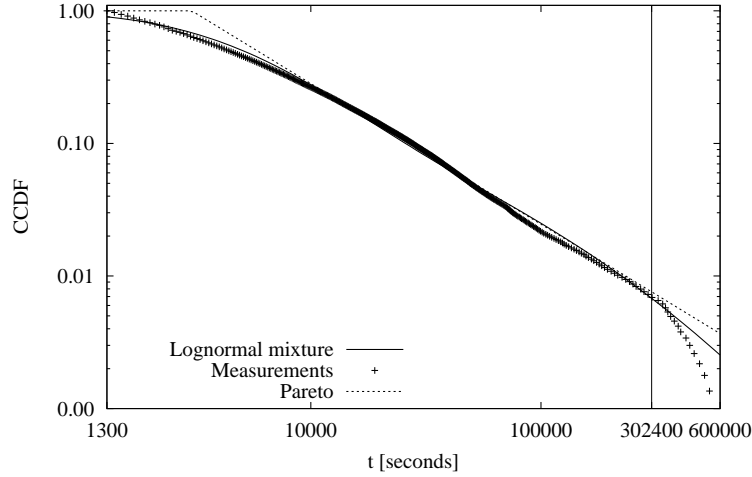


Figure 6.4.: Fitting of lognormal mixture distribution to empirical peer uptime.

Furthermore, the probability density functions (PDFs) of the mixture and the components are shown in Figure 6.5(a); the CDF of the resulting mixture and the components is also provided in Figure 6.5(b). As it can be observed, approximately 70% of the unstable peers depart in less than two hours, while less than 35% of the stable peers depart at the same period and nearly 62% from the mixture.

However, as the driving factors change, the resulting uptime distribution may naturally evolve to a different shape. A general solution to address such unpredictable situations is to approximate the shape with matrix analytic methods (Phase-type distribution) [Neu81]. The latest results in the related research area provide solutions that can efficiently handle even heavy-tailed distributions [HT00], [RDS04]. Though, Phase-type approximations require in general a large number of parameters. Alternatively, one can consider different mixtures of distributions such as the double-Pareto [Mit04] or the double Pareto-Lognormal [RJ04]. We have selected the mixture of lognormal distribution because it fits naturally by considering the evolvement of the underlying process and the heterogeneity of the peers.

Table 6.2.: Parameters of mixture distributions.

Distribution	Parameter 1	Parameter 2	Density probability function
Lognormal Mix. (ratio: $\rho = 0.8$)	$\mu_1 = 8.39$ $\mu_2 = 9.57$	$\sigma_1 = 0.97$ $\sigma_2 = 1.67$	$f_{Mix}(x) = \rho \cdot f_{Lgn}[\mu_1, \sigma_1](x) + (1 - \rho) \cdot f_{Lgn}[\mu_2, \sigma_2](x)$

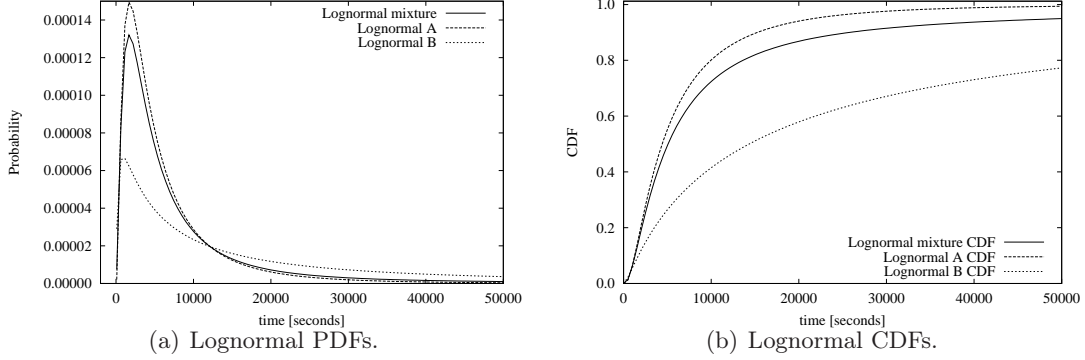


Figure 6.5.: Lognormal Mixture.

6.4.3. Peer Role Cost Model

As it has already been mentioned, P2P systems can either be designed without considering the heterogeneity of the peers (both in physical capabilities and user behavior) and adopting the so-called "*flat*" model, or by taking into account the existing heterogeneity and assigning different responsibilities to each peer (such approaches are usually called "*hybrid*" or "*hierarchical*"⁴³). In the latter case, a well-balanced responsibility assignment mechanisms is mandatory to provide incentives to peers adopting the individual roles. Apparently, many approaches, e.g., JXTA [TAA⁺03], do not consider this issue and (arbitrarily in many cases) select a subset of super-peers. The workload is thereafter unevenly distributed among the peers and almost exclusively assigned to the super-peers, e.g., the maintenance of the overlay network, the consistency of the shared indexing mechanism, etc. More importantly, the expected residual peer lifetime is not considered as an assignment criterion.

On the other hand, a richer and better-balanced role model has been proposed for Omicron [DMS04]. Here four different core roles have been identified: the *Router*, the *Cacher*, the *Indexer* and the *Maintainer*. Within the design of Omicron, peers form clusters and distribute the roles according to their capabilities. For example, low-capability peers assigned the Router role are required to participate only in the message forwarding procedure, while Cachers additionally maintain caches of popular shared items. Peers with higher capabilities can become Indexers and maintain the complete indexing structures assigned to their cluster, while peers assigned the Maintainer role are responsible to maintain the topology of the inter-cluster overlay network, as well as the balanced and stable structure of the cluster itself.

⁴³Chapter 3 provides further details on hybrid and hierarchical overlay network design approaches.

Naturally, the assignment of different roles to peers incurs different costs for the system. For example, in the role model of Omicron, the Routers are required to construct only a small routing table that contains sufficient information to usefully participate in the message routing process. In this case, only a small amount of information must be provided to the peer by the system. On the other hand, an Indexer must copy the complete index of the shared items assigned to the cluster, which has considerably higher cost in system resources.

As it has been mentioned already, there are two different kinds of constraints on assigning a role to a peer, the system and the user constraints. As system constraints are considered the physical capabilities of a peer, e.g., connection type, computation power, energy, etc. As user constraints are considered the behavior and the resulting peer uptime. Since it is less complex to handle the system constraints (i.e., announcing the provided resources and setting bounds on the roles that can be potentially assigned to a peer) our discussion here will consider only the user constraints, alas the peer uptime.

In the general case r different roles might be assumed. However, in a simpler scenario (though adequately expressive and powerful) it can be sufficient to assume only two roles, a lightweight one in terms of required stability that offers the combined functionality of a Router and a Cacher (as described in the Omicron terminology) and a more demanding one that offers the functionality of an Indexer and a Maintainer. The latter role model is considered here, since it suffices to reveal the issues related to the peer burn-in optimization procedure. Let us denote as r_1 the former role and r_2 the latter. The cost for assigning the roles r_1 or r_2 to a peer is c_1 or c_2 , respectively. Deriving from the definition of the roles, it holds that $c_1 < c_2$. This inequality is part of the motivation to consider applying the burn-in mechanism. Afterwards assigning a role to a peer, the system gains a utility from this peer that increases with the time. Let us denote as $f_{u_i}(t)$ the instant utility function of role r_i at time t , which we assume that expresses the "net" profit (the operational cost has been subtracted) for simplicity. For example, the instant utility of role r_1 is the number of messages correctly forwarded towards the destination and the number of successful "hits" in the cache in a unit of time. Similarly, the utility of role r_2 is the number of consistently indexed items available for lookup requests and the maintenance of the optimal topology to guarantee the minimum bounds on routing messages to any destination in the overlay network. Since the utility increases with the time the peer offers the assigned services, the total utility u_i can be calculated by integrating the instant utility function over the time period a role is assigned to the peer.

$$u_i = \int_{t_{assignment}}^{t_{departure}} f_{u_i}(x) \, dx, \quad (6.3)$$

where $t_{assignment}$ is the time role r_i is assigned to a peer and $t_{departure}$ is the time this peer disconnects from the overlay network.

6.5. Burn-in Optimization

6.5.1. Conditions

Let us consider a lifetime random variable X used to model peers' uptime with a continuous distribution function $F(t) = \Pr\{X \leq t, t > 0\}$ and density function $f(t)$. Revisiting the model constructed in Section 6.4.1 the peer uptime is the time until a departure or a failure event occurs; alas, the transition over the transient states towards the absorbing one. *First passage times* analysis [Kul95, pages 162-185] is a suitable method for the investigation of the hazard rate by studying the transition of the process from the transient to the absorbing state.

The *hazard rate* (or *failure rate*) $h(t)$ at time t is defined to be $h(t) = f(t)/1 - F(t)$. The hazard rate is a characteristic of the distribution, capable in many cases of providing more precisely the dynamic features of a distribution than the cumulative, the survival or the density probability function. More specifically, the shape of the hazard rate is the result of the balance between two factors: the attraction of the absorbing state and the general diffusion within the transient space. In general, it has been observed that the shape of the hazard rate is a function of the distance between the starting point and the state of absorption [AG01]. Small distance results to a *decreasing failure rate* (DFR), large distance results into an *increasing failure rate* (IFR), while an intermediate distance leads to a non-monotone hazard rate. It is well known that burn-in is useful for items whose lifetime distributions have initially a high failure rate and then improve over time [BS97]. The set of peers that survive the burn-in procedure has the same failure rate as the original set of peers but shifted to the left. Generally, the decreasing failure rate of the items is the effect of the heterogeneity in their lifetimes.

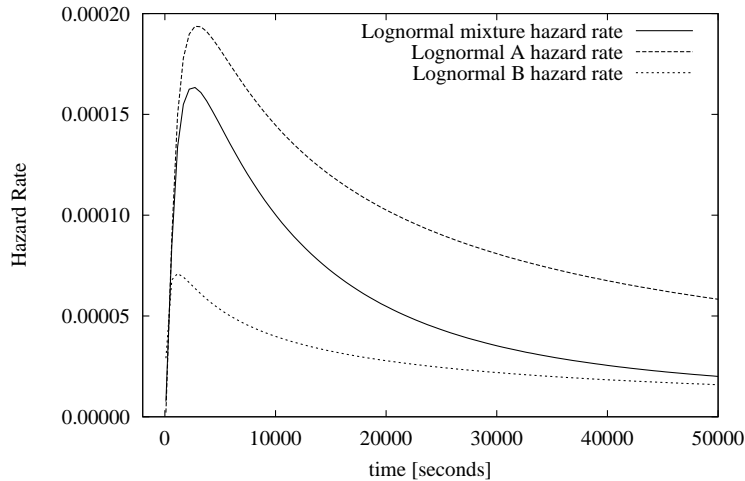


Figure 6.6.: Lognormal Mixture Hazard Rate.

Considering the derived lognormal mixture distribution from Section 6.4.2, we draw the resulted hazard rates in Figure 6.6. By examining the weakest component we observe

a very sharp growth near zero reflecting the fact that the majority of unstable peers perform their initial searching for interesting services. Users can tolerate a short delay to get back the search results. As there is a possibility of not finding the requested resources, unstable users may depart very early. However, their conditional reliability is getting higher as they survive for longer time periods. In contrast, stable users are tolerable and patient to longer delays; the hazard rate is much lower demonstrating their reliability. The mixture distribution combines the characteristics of both components. It is very interesting to observe the limiting behavior of the mixture that converges to the hazard rate of the strongest component (in fact, this is a general property of mixture distributions [BMS93]). It should be noted that the distribution mixture ratio ρ does not influence the limiting behavior; though, it affects the convergence speed. On the other hand, the head of the mixture hazard is mostly affected by the weakest component. Thus, by observing the shape of the mixture, we can conclude that the required conditions for successfully applying the burn-in method are fulfilled.

6.5.2. Criteria

Burn-in optimality criteria can be either *performance* or *cost* based. Performance models rely on basic concepts on which cost models can be further developed. An overview of important criteria is provided in the seminal work of Block et al. [BS97]. In fact, the most basic concept considered in burn-in optimization problem is the *mean residual lifetime* (MRL).

In general, for lifetime distributions that have a bathtub shape it has been reported that the optimal time is before the time the monotonicity of the curve changes. A real valued function $g(t)$ defined on $[0, \infty)$ has a *bathtub* shape if there exist $0 \leq t_1 \leq t_2 \leq \infty$ such that:

$$g(t) = \begin{cases} \text{strictly decreases} & \text{if } 0 \leq t \leq t_1; \\ \text{is a constant} & \text{if } t_1 \leq t \leq t_2; \\ \text{strictly increasing} & \text{if } t_2 \leq t. \end{cases} \quad (6.4)$$

In the above definition, if $t_1 = t_2 = 0$ then $g(t)$ is strictly increasing, while if $t_1 = t_2 = \infty$ then $g(t)$ is strictly decreasing.

The relationship between MRL and the hazard rate have been explored in [Mi95] specifically for cases where the hazard rate has a bathtub curve. In this case, the MRL has an upside bathtub curve. Moreover, it is reported that in the special case where $t_1 = t_2 = \infty$ performance models based only on MRL are not sufficient to estimate the optimal burn-in time and a cost model should be considered. This result holds also when the hazard rate function has a *roller-coaster* shape and the last change has a decreasing slope (Mi [Mi04] investigated the properties of such hazard rate curves and their relation to the shape of MRL).

In fact, the shape of the mixture of lognormal distributions (Figure 6.6) can be described as a roller-coaster curve with decreasing tail. Therefore, it is required to elaborate on the cost model developed in Section 6.4.3 to determine the optimal burn-in time t_b (otherwise, considering only MRL related criteria the optimal burn-in time is infinity).

A useful concept necessary for the following analysis is the *conditional reliability* $R_t(y)$, which is defined to be the probability that a component survives an additional interval of duration y given that it has survived until time t . Thus:

$$R_t(y) = \frac{1 - F(t+y)}{1 - F(t)} = \frac{R(t+y)}{R(t)}, \quad (6.5)$$

where $R(t) = 1 - F(t)$ denotes the reliability (survivability) function.

6.5.3. Mission Time Estimation

Generally speaking, the burn-in technique is employed to enhance the reliability by eliminating weak items from a population of items having heterogeneous lifetimes. In the original definition of the term, the procedure includes an accelerated life test that is applied to the components incurring (sometimes significantly) additional cost. However, in the case of P2P overlay networks, accelerated life tests are not possible and the mechanism merely let peers survive for a *burn-in time* t_b , which maximizes the probability that a peer will survive for an additional time t_m , the so-called *mission time*. Hence, the cost of the procedure is merely the lost utility during time t_b when the tested peer is not utilized effectively by the system.

The problem we seek to optimize is formulated as follows. Assume that assigning a role to a peer incurs a cost c_a , which depends on the network quantities such as the population, the advertised items, the number of neighbors required, etc.; nevertheless, this cost is independent of the behavior of the individual peer. In order to design a cost effective system, the utility gained by a peer must be greater than the cost required to maintain the necessary changes. This raises a constraint on the minimum utilization time $t_m = t_{\text{departure}} - t_{\text{assignment}}$, the so-called *mission time*. In fact, for the efficient assignment of role r_i it should hold that :

$$c_a \leq u_i. \quad (6.6)$$

Using (6.3) to replace u_i we get:

$$c_a \leq \int_{t_{\text{assignment}}}^{t_{\text{departure}}} f_{u_i}(x) \, dx, \quad (6.7)$$

where $t_{\text{assignment}} = t_{\text{join}} + t_b$. For simplicity, if we assume that the utility rate is constant over time, i.e., $f_{u_i}(t) = u_c$, Inequality 6.7 becomes:

$$\begin{aligned} c_a &\leq (t_{\text{departure}} - t_{\text{assignment}})u_c &\Rightarrow \\ c_a &\leq (t_{\text{departure}} - t_{\text{join}} - t_b)u_c &\Rightarrow \\ \frac{c_a}{u_c} + t_b &\leq (t_{\text{departure}} - t_{\text{join}}). \end{aligned}$$

The above inequality implies that the peer should survive for at least $\frac{c_a}{u_c} + t_b$ time units. The minimum mission time t_m is then

$$t_m = t_{\text{departure}} - t_{\text{join}} - t_b = \frac{c_a}{u_c} \quad (6.8)$$

6.5.4. Cost-based Optimization

Having estimated the mission time t_m , we need to consider a cost-based criterion to optimize the burn-in procedure, as it has been discussed in Section 6.5.2, since MRL alone is not sufficient for this case.

The cost of not utilizing the peer immediately at the join time but waiting for the burn-in time t_b is c_l , defined as:

$$c_l = \int_{t_{join}}^{t_{join}+t'_b} f_{u_i}(x) dx, \quad (6.9)$$

where t'_b , $0 \leq t'_b \leq t_b$ is the time of not utilizing the peer.

The cost function of the burn-in procedure is given by:

$$E[C_b(t_b)] = u_c t_b + c_{bf} F(t_b) + c_{mf} [F(t_b + t_m) - F(t_b)], \quad (6.10)$$

where the first term represents the lost utility c_l for the burn-in period as calculated in (6.9) for $t'_b = t_b$. The second term is the cost to recover from a failure during the burn-in period t_b assuming c_{bf} cost units per time unit and the last term is the cost of failure during the mission time t_m assuming c_{mf} cost units per time unit. Since $c_{bf} \ll c_{mf}$, we can safely ignore the second term, getting a simpler cost expectation function:

$$E[\tilde{C}_b(t_b)] = u_c t_b + c_{mf} [F(t_b + t_m) - F(t_b)], \quad (6.11)$$

where $F(t)$ is the CDF of the lifespan. Using (6.5), Equation (6.11) can be written as:

$$E[\tilde{C}_b(t_b)] = u_c t_b + c_{mf} R(t_b) [1 - R_{t_b}(t_m)]. \quad (6.12)$$

The minimum cost occurs at t_{min} so that:

$$E[\tilde{C}_b(t_{min})]' = 0, \quad (6.13)$$

which is the optimal burn-in time considering the burn-in cost. It should be noted that for certain values of the cost and utility parameters as well as the reliability function, one might compute that the minimum cost is observed for $t_b = 0$. In such scenarios where for maximum MRL the burn-in time must be $t_b = \infty$ while for minimum burn-in cost the burn-in time must be $t_b = 0$, it is necessary to explore another criterion comparing the cost and utility of the system with and without burn-in. Nevertheless, such scenarios do not apply in the empirically observed P2P systems.

6.5.5. An Example Use Case

Aiming to a more comprehensive understanding of the derived results, we provide an example use case. In this example a P2P overlay is explored. It is assumed that the cost to assign a demanding role to a peer is $c_a = 10000$ and the constant utility is $u_c = 2$ per second. The cost of failure within the mission time is $c_{mf} = 60000$. From (6.8) we get that the mission time is $t_m = 5000$ seconds. Applying these values to (6.13) we derive

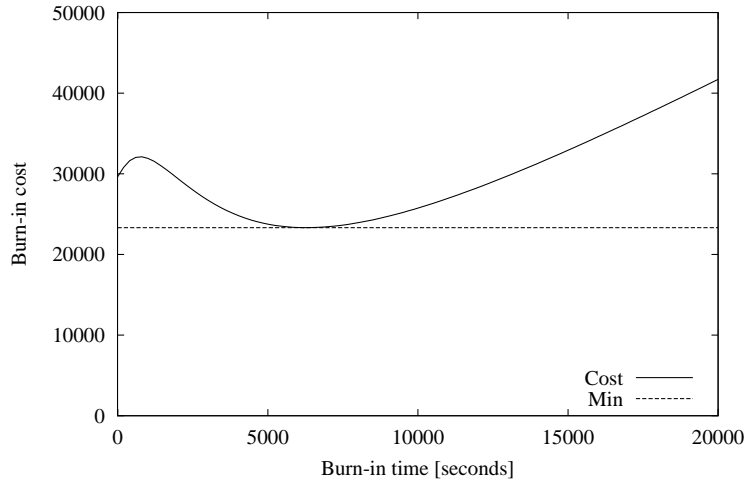


Figure 6.7.: Burn-in cost.

the solution for the optimal burn-in time under the assumed parameters. In particular, the solution for the optimal time of this example is $t_b = 6277.52$ seconds and using (6.12) the expected cost for the burn-in period is calculated ($E[\tilde{C}_b(6277.52)] = 23320.7937$). A graphical representation of the cost as a function of the burn-in time t_b is provided at Figure 6.7.

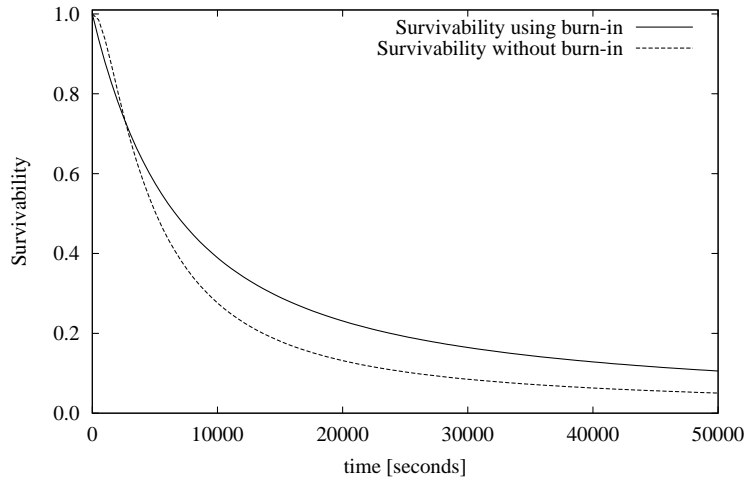


Figure 6.8.: Survivability comparison.

After calculating the required parameters, the focus is shifted towards how to examine the benefits of applying the burn-in method in the resulting survivability of the P2P system. Figure 6.8 shows the achieved survivability of the peers assigned a critical role, with and without burn-in appliance. The advantage of the method is clearly displayed. It should be clear though, that the burn-in appliance does not make peers better; the procedure does

not affect the behavior of the peers. Instead, by capitalizing on conditional probabilities, components that survive burn-in have a higher probability of having longer lifetime than components without burn-in.

6.6. Cluster endurance

The weak cluster endurance $E_{CW}(t)$ has been defined in Section 4.3.1 as the probability that at least one peer will survive after t time. Having estimated a realistic survivability function for peers based on empirically observed systems, an estimation of the cluster endurance is demonstrated in this section.

Assume that a cluster consists of 5 peers. Their observed up-time is provided in Table 6.3. Based on the required burn-in time calculated in the previous section, it can be noted that two of them have survived long enough to safely be assigned demanding roles.

Table 6.3.: Peer up-times configuration example.

Peer	Observed up-time	Survived burn-in
P_A	500 seconds	
P_B	1000 seconds	
P_C	6000 seconds	
P_D	11000 seconds	✓
P_E	11000 seconds	✓

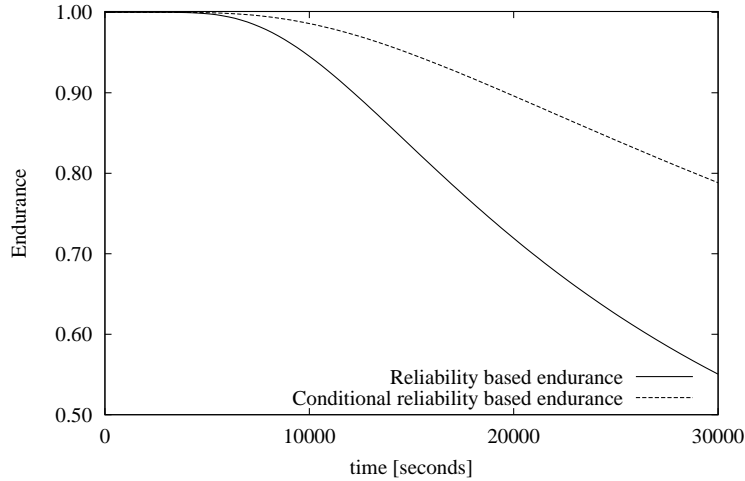


Figure 6.9.: Cluster endurance distribution.

The endurance of this particular cluster is shown in Figure 6.9. There, a comparison between peer reliability and conditional peer reliability based endurance estimation is provided. It can be noted that the conditional reliability based estimation provides a higher endurance value, which reflects better the expected endurance since it utilizes better

the available information. This is particularly important since as it has been observed the uptime distribution does not follow a Poisson distribution; thereby, it cannot be considered as memoryless.

A usual metric to estimate the reliability of a service in the telecommunications sector is to calculate the expected elapsed time so that the service is available with probability 0.99999 (the so-called "five nines" rule). Thus, it is essential to calculate the time t_{fn} such that $E_{CW}(t_{fn}) > 0.99999$. This time is the expected time that at least one member of the cluster will survive with probability 0.99999. Using Equation 4.2 that provides the weak cluster endurance and setting the reliability $f_i(t)$ of each peer to the conditional reliability formula provided in Equation 6.5 we calculate $t_{fn} \approx 2,748.88$. Similarly, using the reliability function we get that $t_{fn} \approx 2,701.89$. In both cases it is assumed that the peer uptime distribution is given by the lognormal mixture distribution defined in Section 6.4.2. The distribution parameters are given in Table 6.2. Figure 6.10 shows a "magnification" of the top left area of Figure 6.9 to graphically illustrate the "five nines" concept.

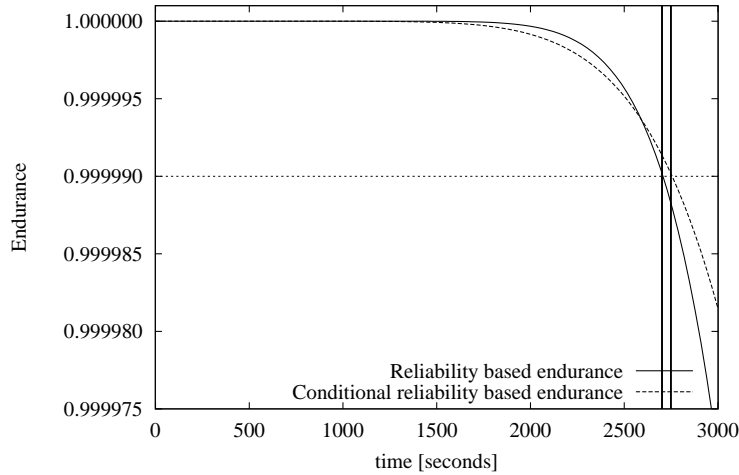


Figure 6.10.: Expected time for endurable clusters ($E_{CW}(t_{fn}) > 0.99999$).

6.7. Summary

In order to find the optimal time to assign a critical and demanding role to a peer, a number of issues have to be addressed. The most challenging question was to define the peer uptime distribution. While there are a plethora of empirical observations, researchers still debate the appropriate fitting distribution. In fact, by considering similar discussions on different Internet quantities such as the size distribution of Web files, it is observed that seemingly trivial variations on generative models can result in different distributions. Moreover, we investigated and collected the core factors that define the boundary within which the empirical observations can be trusted. By primarily considering the heterogeneity of the peers and the generative multiplicative process that drives the underlying process of peer's

lifetime, we provided some meaningful and natural interpretations of a representative empirical curve. Apparently, the multiplicative process fits quite adequately with the case of swarm-based file sharing services where many small factors influence its progress. However, as the driving factors change, the resulting uptime distribution may naturally evolve to a different shape that may be better approximated by different methods (e.g., matrix analytic techniques).

Elaborating on the derived hazard rate functions of peer lifespan, we examined whether it is meaningful to apply the burn-in method on peers. Thereby, the necessary conditions have been identified and the appropriate criteria have been selected. Further, a stochastic framework has been constructed to compute optimally (minimum burn-in cost) the parameters of interest, viz. the optimal mission time and burn-in time. The results have been applied to a parametric example of P2P system where the identification of peers with statistically higher survivability allowed the assignment of critical roles to them, leading to a globally more stable overlay network. Moreover, the estimation of the expected cluster endurance is demonstrated based on peer reliability and conditional peer reliability methods, where the improved results of the latter method are observed.

The findings of this chapter (especially the parametrical description of peers' uptime) are important for the stochastic analysis of the P2P overlay networks.

Open Architecture Simulation Framework

Fast is fine, but accuracy is everything.

XENOPHON

Conceptual outline.

Having described the architectural design of Omicron as well as the modeling and analysis of the relevant communication mechanisms, it is necessary to evaluate the accomplished system performance and compare it with existing alternative approaches. However, before providing the performance evaluation, the architectural description of the developed simulation-based evaluation framework itself is supplied. Since there was no suitable simulator available to meet the raised requirements, it was necessary to develop a new tool. The open software architecture of our tool is described both from functional and compositional point of view. The tool has been designed to meet the needs of a plethora of P2P overlay networks and not merely the particular needs of Omicron. However, its design is influenced by the rich role model of Omicron, which constitutes an excellent approach to effectively implement other systems (even systems that do not consider the heterogeneity of the peers). In fact, both Chord and Gnutella have been implemented within the simulator. The tool can export the simulation results in multiple formats suitable either for creating meaningful graphs or detailed descriptions of the running simulations that may be visualized by employing appropriate external tools.

This chapter is organized as follows. The need for a novel simulator appropriate for P2P overlay networks is motivated in Section 7.1 and a brief overview of the state-of-the-art on P2P overlay simulators is supplied in the same section. Subsequently, the architecture of the simulator is described in Section 7.2 including the functionality, the components and additional crucial information. This is followed by the description of the underlying network model in Section 7.3. Then, the Omicron role model is adopted and extended to meet the needs of several other P2P overlay networks in Section 7.4, followed by the description of the message handling procedure in Section 7.5. Finally, the chapter is summarized in Section 7.6.

7.1. Motivation

Up to this point the architecture of the P2P overlay network and the relevant communication algorithms and mechanisms have been described, modeled and analyzed. Therefore, it is crucial to evaluate and compare their performance with existing alternatives in an environment as pragmatic and viable as possible. The complexity of the overlay operations and the great multitude of the objectives need to be provably assessed using simulative and (to a certain degree) analytical techniques before proceeding to the expensive step of system deployment. Thus, the most adequate approach for this performance evaluation is to simulate the system using a sufficiently detailed model. Moreover, equally expressive models must be developed for alternative systems. As it has been discussed in Chapter 3, the design space of P2P overlay networks relatively large. Therefore, there is a need for a general and powerful tool that can supply both rich design capabilities, as well as concrete useful functionality to realize the plethora of networks. In addition, it should be efficiently implemented in order to be capable of supporting simulation experiments of relatively large sizes (several thousands of peers).

Nevertheless, developing a simulator is a process that poses a number of design and implementation issues. The depth at which a simulator models the problems should be sufficient to get valid results. Additionally, it should provide sufficient functionality as well as be able to extend and adapt its design to cover new concepts. However, when models become large in size they require extensive support in memory and computation power to perform the experiments. To balance out these trade-offs at the level of operational details an effective design is required.

On the other hand, modeling P2P systems poses certain requirements on the ability to capture the involved complexity. More specifically, effective mechanisms are required to model the user behavior as well as the details of the communication protocols at the overlay level. Further, detailed modeling of the underlying physical network is necessary at certain occasions. To enable the simulation of complex approaches, advanced concepts such as clustering mechanisms and role assignment should be supported as well.

Several simulators have been examined as potential candidates that could be used to extensively assess the characteristics of diverse P2P approaches. Ns-2 [FV00] had been initially considered mainly because of the great popularity of the tool and the provided support. While Ns-2 is the default option for transport and lower network protocols of the Open Systems Interconnection (OSI) model, it proved to be inefficient for P2P systems. The detailed model of the network routers and the transmitted packets, made the simulation of large size P2P systems very demanding. Moreover, the duality in the language requirements (both C++ and Tcl) made it less attractive.

J-Sim (formerly known as Javasim) [Tya02] has been another evaluated candidate. Its layered-design appears to be a promising alternative that can enable the simulation of larger systems. Initially, J-Sim had been used to provide the underlying network model but due to the large resource consumption it has been replaced by a more lightweight solution. A different approach, the Neurogrid Simulator [Jos03] has been designed specifically for P2P systems so it offers a more efficient model. However, the fact that it has been initially

designed for a specific system (Neurogrid) and the lack of sufficient documentation limits its potential as a general-purpose P2P simulator.

Ptolemy [BCD⁺03] is a general-purpose simulator that offers a large variety of simulation models. Though, it is not a lightweight solution appropriate for large-scale P2P systems. Finally, some other simulators have been briefly examined, such as 3LS [TD03], Desmo-J [PLC00], Simjava [HM98], myns [Ban02]. For different reasons, they have failed to meet the posed challenging requirements.

7.2. Simulator Design

7.2.1. Functionality Layers

In order to address effectively the complexity of P2P systems, an analysis of their functionality in different layers has been performed to divide them in distinguishable parts. Their functionality can be effectively represented by the following three layers ⁴⁴:

1. *User behavior layer.* This layer captures the actions taken by the users. This might include their behavior using advanced services and express their co-operativeness or their inclination to become free riders (a well-known phenomenon in the context of P2P systems). In addition, this layer captures important information for the reliability and stability of the P2P overlay. It is very important to simulate accurately the rates at which peers join and leave P2P overlay networks. Based on this information prediction algorithms can be developed to optimize the operation of the overlays and the usage of the related resources.
2. *Overlay protocol layer.* Overlay protocols (are application layer protocols for constructing the overlay networks and performing the related operations) are included in this layer. More specifically, this layer consists of the communication protocols of the overlay networks and the corresponding algorithms such as message routing, dissemination of system-related information (e.g., estimated peer reliability), maintenance of the overlay structure, etc. This layer is the most significant one with respect to the context of this thesis.
3. *Network transmission layer.* This layer captures the details of the physical network (e.g., latency and bandwidth) and the related communication protocols that are located at the four lower layers of the OSI model. Although in many cases the details of this model are not of high importance for the researchers of P2P systems, they can play an important role (especially in cases where network QoS is of great importance such as media streaming within P2P overlays). This layer is responsible for modeling the details of delivering packets between peers.

Figure 7.1 shows the functional architecture of the simulator. It is designed in a way that can naturally capture the needs of the P2P systems as they have been analyzed. The *Agent* layer represents and models user behavior, the *Overlay* layer includes the details of the

⁴⁴ A similar functionality separation is proposed in [TD03].

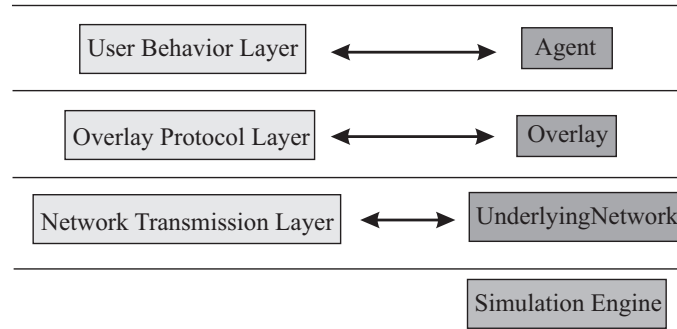


Figure 7.1.: Functionality layers.

P2P protocols and the *UnderlyingNetwork* is the layer that enables the attachment of a variety of network models that can capture the physical network properties in a wide range of detail levels. Finally, the architecture includes a core component of the simulator, the *SimulationEngine*, in addition to the functional layers. The simulator has been designed in a way that a great multitude of simulation engines, e.g., event-based or process-based can be attached to perform the simulation. A default simulation engine is provided, though users can exchange it with customized ones that may be more appropriate for certain scenarios.

7.2.2. Components

The aim of developing this tool is to provide a general-purpose P2P simulator, which is not dedicated to any specific P2P architecture or system. For this reason it has been designed following a framework-like approach. As a first step, many different P2P overlay architectures (structured, unstructured, flat, hierarchical, hybrid) have been analyzed to capture their requirements.

Following this step, the most important components have been identified and developed as well-defined entities in the software architecture of the simulator. The framework of the simulator is based on the concept of plug-ins. More specifically, a well-defined interface is supplied for each identified core component to offer the required functionality. Additionally, for each interface either a default implementation (in cases where it offers general-purpose functionality suitable for any P2P approach) or an abstract base implementation with general functionality is provided. Users can customize the concrete implementation of each interface either by replacing the default component implementation or by providing specific extensions to the abstract classes. The process is straightforward since there is a well-defined interface that can be seamlessly referenced in the implementation of other components; changes are only locally required inside the component. As an example of a component of this category, consider the *Simulation-Engine*. Users can replace the default event-based engine with a process-based engine. An abstract base implementation with general functionality is provided in cases where no default implementations can be proposed to fulfill the needs of each P2P overlay network. The base implementation has to be extended to satisfy the needs of the particular system. The *Agent* class is an example

component of this category that has to be extended.

The interaction among the default or the customized components takes place through their well-defined interfaces, making the interaction process agnostic of implementation details. Figure 7.2 illustrates the aforementioned design patterns of the core components.

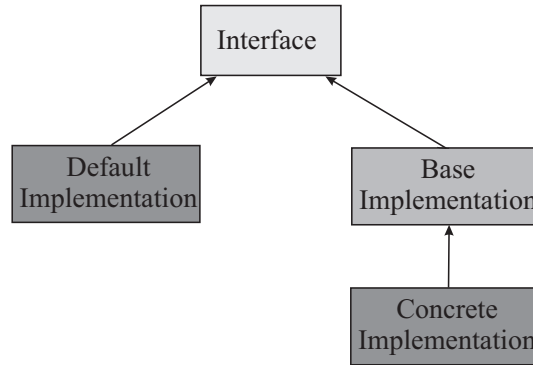


Figure 7.2.: Component design pattern.

The extended analysis of many well-known P2P systems provided a comprehensive understanding of their structure. The most important identified concepts that have been captured by the architecture of the simulator are supplied in the following list.

- *Peer*. It represents the peer itself encapsulating all the related components.
- *GUID*. Every peer can be uniquely identified using a globally unique identifier (GUID). In many cases, no central authority exists to guarantee the uniqueness of the identifiers so efficient distributed algorithms are employed.
- *Overlays*. They encapsulate the components that are related to the construction of the overlay networks. In many cases peers participate in many sub-overlays (especially in cases of hierarchical approaches).
- *Cluster Map*. Many systems introduce the concept of clusters of peers where peers are grouped together with respect to certain constraints, user interests or network proximity-based requirements. Cluster Maps are entities that summarize the membership information of each cluster.
- *Routing Table*. Each peer maintains a number of neighbors in order to construct the overlays. The Routing Tables encapsulate this information. The scope of each Routing Table is the corresponding overlay network.
- *Roles*. In many cases, having identical responsibilities for each peer can lead to inefficient systems. Many approaches propose the identification of roles appropriate for different peers in order to overcome this problem.
- *Message Dispatcher*. Although this component is not very important from a conceptual point of view, it plays an important role in the correct and valid operation of the peers. Peers that participate in multiple overlays and hold a number of different roles require a mechanism to direct the incoming messages to the appropriate

receivers.

- *Messages*. A large number of different messages may exist in P2P systems. They are vital components to develop the distributed nature of the P2P systems.
- *Protocols*. This concept represents the different overlay protocols. It is helpful in the modeling of the systems. A protocol is constructed by a well-defined set of messages.
- *Documents*. Documents represent items of interest that can be shared between peers. They are identified by GUIDs.
- *Index*. Indices are structures that summarize the location of the documents in the P2P system. They can be detailed structures or can be modeled using appropriate probabilistic methods to simulate their operation.
- *Cache*. Caches are document repositories of limited size. They can significantly improve the performance of the overlay operations, e.g., in cases of Zipfian query distribution.
- *Link*. Links encapsulate the details of the physical connections among peers. They can include complex models for the latency and the bandwidth or simple models that offer certain functions, e.g., in-order message delivery.
- *NetworkWrapper*. NetworkWrappers are general objects that adapt and hide the details of the underlying network from the Overlay layer. They also encapsulate Links.

Figure 7.3 shows the architectural components of each Peer. The multiplicity of the particular entities is demonstrated with overlapping rectangles.

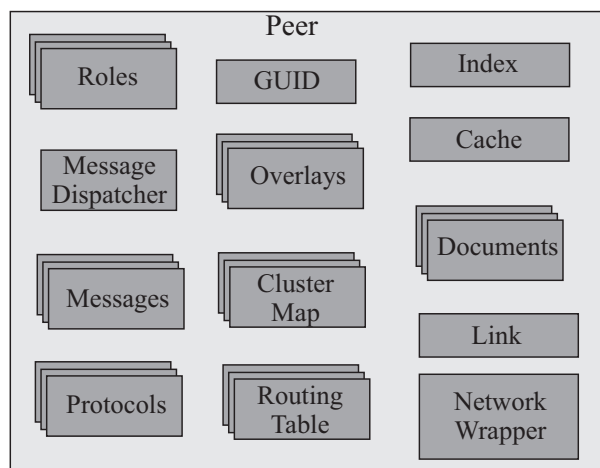


Figure 7.3.: Core peer components.

7.2.3. Simulation Engine

A framework approach has been adopted to allow the easy replacement of the default simulation engine with customized engines. The requirements for the simulation engine are efficiency and sufficient functionality. Figure 7.4 provides the software design of the default simulation engine. The *SimulationFramework* is a general-purpose component that is called by the *Application* to create certain scenarios (build nodes, agents, topologies, etc.), start them and process the results of the experiments. The *Simulator* is an interface that provides vital information about the status and the progress of the simulation. The *Scheduler* enables the insertion of the active components into the simulator that can generate events. It ensures the correct execution of those generated events in order to provide valid experiments. The duration of each experiment is controlled by the scheduler and the parameters defined by the *Application*. *Peers* add the generated events indirectly via the *NetworkWrapper* interface. The events themselves are modeled by the so-called *Event* class. They provide information such as scheduled execution time, content of the messages, destination and originator. An efficient implementation of an ordered *Queue* is used to store the generated messages and make them available for execution. Moreover, Agents can generate *Actions* that are also added in the Queue of the Scheduler. Actions share many similarities with the Events.

The default implementation of the simulation engine is single-threaded with respect to the model actions in order to avoid the complexity of multi-thread programming. Costly I/O operations can be performed by different threads to improve the overall performance.

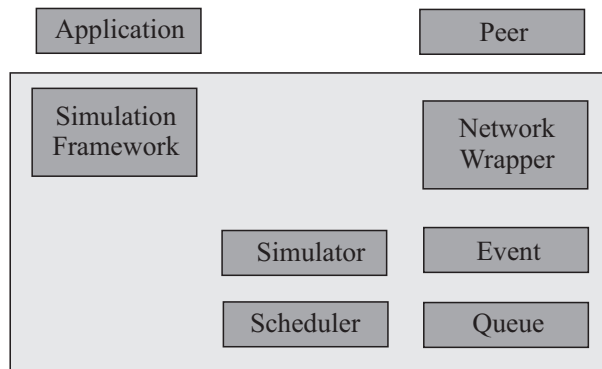


Figure 7.4.: Simulation engine.

7.3. Underlying Network Support

As it has been already described, the architecture of the simulator enables the easy adaptation of many components to fulfil a variety of functionalities. Holding the same axiom for the underlying network layer, a general interface (*NetworkWrapper*) has been defined to encapsulate the details of the networking layer. A lightweight default implementation

is provided that offers an adequate model with minimum complexity and demands in resource requirements. The characteristics of this model are given in the following list:

- *Reliable communication*: Taking into account the fact that most of the deployed P2P systems employ TCP as a transport protocol to exchange messages, this model provides reliable communication in on an end-to-end basis. This means that no packets will ever be dropped because of buffer overflows in routers.
- *Peers may depart arbitrarily (fail)*: Peers may depart without prior notification. This is the only type of failure that can take place in this simple model. Network connectivity is assumed.
- *In-order message delivery*: Messages that belong to the same session are guaranteed to be delivered in-order. Again, this follows from TCP like connection modeling.
- *Probabilistic end-to-end delay*: In this model links are entities that connect two peers (end-to-end) as illustrated in Figure 7.5. The details of the physical network are hidden in this model.

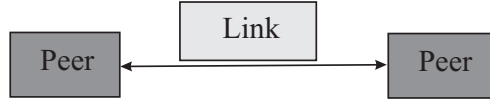


Figure 7.5.: Link modeling.

Aiming to increase the ability to correctly set the end-to-end delay without modeling a very detailed and complex underlying network, the following model has been developed. The concept of *SubNet* has been introduced to model network Autonomous Systems (ASs). Peers that belong to the same SubNet are assumed to be close in the network (few hops away) and thus, the transmission delay among these peers is assumed to be short as well. On the other hand, peers belonging to different SubNets have longer delay. The delay in a packet switched network (i.e., IP based) is a function of the physical distance which is a fixed delay and a is mostly effected by the network congestion and the length of the constructed queues in the intermediate routers. The latter factor is dynamic and it may be modeled stochastically.

The proposed model is transforming the peer delay into a two-dimensional Euclidean space where each peer is mapped and assigned two coordinates. Naturally the distance between two peers P_A and P_B (represented as Euclidean points where their coordinates are (x_A, y_A) and (x_B, y_B) , respectively) is given by:

$$distance_{AB} = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}. \quad (7.1)$$

Thus, the delay between P_A and P_B is a stochastic function of their distance:

$$delay_{AB} = \alpha \cdot distance_{AB} + q(distance_{AB}), \quad (7.2)$$

where $q(x)$ is the congestion delay and α is a parameter that can regulate the distance effect.

The SubNet concept is illustrated in Figure 7.6. There, the peers are physically connected to three different ASs. Their mapping to the abstract Euclidean space is shown in a way that their relative distance can be adequately approximated.

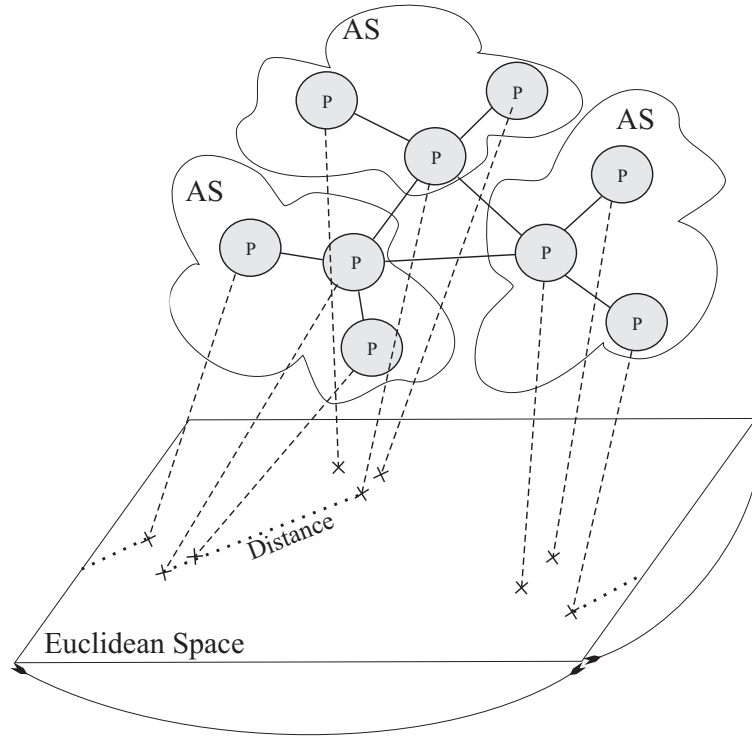


Figure 7.6.: Euclidean distance abstraction.

7.4. Roles

A novel characteristic of the simulator is the ability to support multiple roles for each peer in an effective way. Roles represent responsibilities for certain operations, such as routing or indexing. In order to achieve this an appropriate abstract framework is necessary. A common role has been identified for each peer independently of the selected system to be simulated. It is the *Connector* role, which provides the establishment of the inter-peer connections (not necessarily in a TCP way). Its detailed description is given in Chapter 5. The Connector is responsible to handle the creation and tear-down of each inter-peer connection. Moreover, it offers the ability to check the validity of the connections using ping/pong messages. Furthermore, the Connector offers the means to dynamically update the roles "installed" on each peer. Finally, it dynamically updates the local ClusterMaps as they evolve over time.

Table 7.1 provides the identified roles as they appear in the different node types of three P2P networks. These P2P networks are representative systems of different design approaches. JXTA is a hierarchical system, Chord is a non-hierarchical P2P approach and Omicron is a hybrid one. Five core operations are shown: *Routing* (RT), *Caching* (CH),

Indexing (IX), *Maintaining* (MN) and *Connecting* (CN). The advantage of the role-based approach is the simple reconfiguration of the responsibilities and the re-usage of the implemented modules in multiple ways. For example, users can share the same Indexing mechanism between Chord-based and Omicron-based experiments. Alternatively, simple JXTA peers can dynamically evolve to Rendezvous by installing additional roles.

Table 7.1.: Simulator roles and overlay networks.

Node type	Operations				
	RT	CH	IX	MN	CN
JXTA Rendezvous	✓	-	✓	✓	✓
JXTA Peer	-	✓	-	-	✓
Chord	✓	-	✓	✓	✓
Omicron Router	✓	-	-	-	✓
Omicron Cacher	✓	✓	-	-	✓
Omicron Indexer	✓	-	✓	-	✓
Omicron Maintainer	✓	-	✓	✓	✓

7.5. Message Handling

Messages are the entities that encapsulate the information to be exchanged among peers to implement the overlay protocols. In complex systems such as JXTA, peers can participate in multiple overlay networks, e.g., per peer group. Moreover, they can be assigned multiple roles, e.g., Router, Maintainer, etc. Peers communicate with their neighbor peers for each assigned role in every overlay abstraction. Each message carries a number of fields that can augment this process to handle efficiently the large number of potential receivers. The following common fields are included in each message:

- *Style*. The style determines the way a message is transmitted to the destination. There are two different styles. The *recursive style* where the message is forwarded to the most "promising" neighbor. Subsequently, this neighbor forwards the message to the most promising neighbor of his until the message arrives at the final destination. The second style is the *iterative communication style*. Also in this case, the message is forwarded to the most promising neighbor. However, the selected neighbor replies to the original peer with the address of the most promising of its neighbors instead of forwarding the message directly. Then the original peer is responsible to contact the new peer and get a new address until it reaches the final destination. Although the second communication style is more costly in terms of latency compared to the first one, it has the benefit that peers can check the progress of forwarding messages themselves, thus, avoiding the problems that might appear with the presence of malicious peers.
- *Type*. The type of the message determines the functionality that this message serves. For example, it can be a maintenance or a routing message.

- *Name*. The name discriminates the messages and enables the selection of how to react to each one.
- *Scope*. The scope of a message determines the overlay through which this message should be delivered to the destination. It might be possible that a destination node might be reachable through different sub-overlays at different cost.

Figure 7.7 displays the common structure of each message. Moreover, this figure shows how messages are handled when they arrive at a peer. The Message Dispatcher checks the Type of the message and it passes it to the appropriate Role. Subsequently, the Role examines the Name of the message in order to process it. After this, it performs the appropriate local actions and it examines the scope of the message in order to select the appropriate Overlay to forward or to reply to this message.

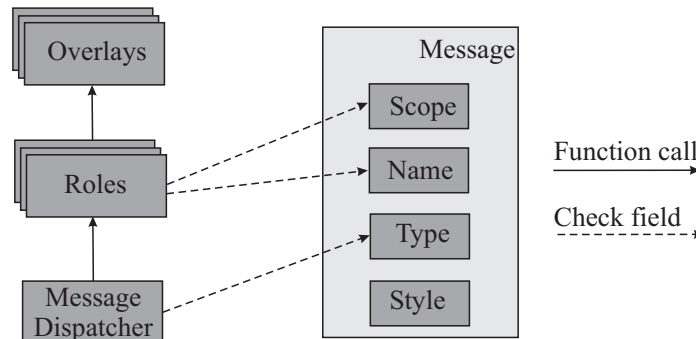


Figure 7.7.: Message handling.

7.6. Summary

Though simulation alone is not sufficient to guarantee successful deployment of P2P systems over the Internet, it is a mandatory step in the development cycle in order to reduce the deployment cost of unforeseen scenarios.

A well-developed simulator should carefully address the need for functionality and the achieved performance. Moreover, an extendable and adaptable design is necessary to enable the integration of particular systems with the general-purpose mechanisms offered by the simulation framework.

The described tool has been designed to simulate network operations particularly in the case of P2P systems. The role-based approach increases its suitability for a variety of overlay network designs. The layered-based approach enables the attachment of various underlying network models to fit the level of desired detail, e.g., J-Sim.

Furthermore, performed experiments show the efficiency achieved in terms of resource requirements (memory and computation) for sufficiently large P2P networks. The high adaptability of the simulator to a variety of simulation engines increases its suitability to a vast number of modeling approaches.

Currently four P2P systems have been ported for simulation: Gnutella, Chord, JXTA and Omicron. Simulation is a preferable approach to compare P2P networks since identical environmental conditions can be assured to obtain objective and comparable results. It is the focus of the following chapter to provide the collected results that have been produced using the described simulation tool.

Evaluation Experiments

The test of any man lies in action.

PINDAR

Conceptual outline.

Throughout this thesis, a novel architecture for P2P overlay networks has been designed, including a comprehensive set of algorithms and mechanisms. The targeted requirements of this system have been analytically evaluated for several cases. However, in order to assess more accurately the performance of the system in more complex scenarios, simulation has been applied to validate the approach. The experiments have been performed using the simulation framework described in Chapter 7. Most of the them focus on the performance of Omicron. Nevertheless, it is not merely the absolute performance that matters, but the performance compared to other approaches. Therefore, similar experiments have been performed evaluating Chord (wherever this is meaningful) in order to provide a reference point for comparison. Moreover, in order to demonstrate the general applicability of some mechanisms (i.e., the investigated caching mechanism), Chord has been selected as the assumed network topology for these experiments. Some further results gathered by additional experiments are provided in Appendix D.

This chapter is organized as follows. Section 8.1 describes the general settings of the simulation experiments. Afterwards, the accomplished scalability is quantitatively evaluated in Section 8.2. The ability of Omicron to efficiently exploit the network proximity is discussed in Section 8.3. Next, the focus of Section 8.4 lies on the routing workload distribution where several quantities are evaluated. Following the evaluation of the routing mechanisms, Section 8.5 investigates the performance of multiple algorithms on their ability to sample adequately the endurance of each cluster. Then, the performance improvement when the proposed caching mechanism is used is evaluated in Section 8.6. Finally, the chapter is summarized in Section 8.7.

8.1. Experiments Description

The purpose of this chapter is to comprehensively evaluate the proposed P2P overlay network using simulation results. Its goal is to quantitatively evaluate the investigated system and supplement the accomplished theoretical analysis. The evaluation process is based merely on simulation experiments. Moreover, analytical approximations of the evaluated quantities are carried out whenever possible in order to provide a direct comparison for a range of input values.

The simulation experiments have been performed using the general purpose discrete event simulator for P2P overlay networks presented in Chapter 7. The reported experiments are based either on Omicron or on Chord. The characteristics of interest include the generated traffic to locate and deliver the requested information, the distribution of the load among the participating peers and clusters, the efficient use of the underlying network, etc.

The size of the evaluated P2P networks is ranging from 100 to 130,000 peers⁴⁵. During the performed experiments the size of clusters ranges between 2 and 16 peers. The degree of the de Bruijn networks is mostly set to 2 and 4 connections per node. While it is possible to set it to any arbitrary integer, the interest of our research lies in fixed, low degree de Bruijn networks.

Networks are constructed as peers join in the so-called bootstrap phase. When the targeted population is reached, peers perform operations and statistics are collected. Events are either randomly or periodically generated according to the needs of the particular experiment. Several random number generators are used to provide the appropriate input rates, e.g., uniform, normal, lognormal, Zipfian, etc.

8.2. Scalability

Scalability is the most critical requirement for most P2P applications. In terms of core overlay network operations, the cost of `LookupRequest(s)` is the metric we use to evaluate the scalability of the network. It should be noted that the cost both for `LookupRequest(s)` and `PublishRequest(s)` (as they are described in Chapter 5) is identical. In the rest of this chapter we refer to such messages as *queries*. The focus of the experiments is mostly on the way the communication cost increases as the network grows in size.

8.2.1. Impact on Shortest Path Distribution

In this section the quantity of interest is the distribution of the shortest paths between nodes. This investigation reveals the relationship between the average shortest path and the network diameter. Loguinov et al. [LKR03] provide an approximation formula of

⁴⁵It should be noted that not many widely available simulators have such an ability to simulate network models of that scale and model detail.

the asymptotic distribution (PMF) of shortest paths in de Bruijn graphs:

$$p(n) \approx \frac{k^n}{N} - \frac{k^{2n-1}}{N^2}. \quad (8.1)$$

Here N is the order of the de Bruijn network and k is the degree of each node. Unfortunately, the distribution of de Bruijn distances $d(x, y)$ between two nodes x and y is very complicated and there is no closed-form expression for its PMF [SR94]. Equation 8.1 is exact when the diameter $D \leq 3$ and very close to the real value for larger graphs.

In the related experiments many different orders of Omicron networks have been evaluated. Nevertheless, we provide only the results of two representative experiments to give a better idea of the shortest path PMF. For these experiments, the size of the Omicron network is set to $N_1 = 16,384$ and $N_2 = 131,072$ and the average cluster size is $\bar{K} = 8$. The degree of the network is set to $k = 2$. Therefore, the order of the de Bruijn network in these two scenarios is $M_1 = 2048$ and $M_2 = 16384$, respectively (note that $M = N/\bar{K}$). Figure 8.1 provides the graphical representation of the results. The analytical approximation given by Equation 8.1 is drawn as a continuous function to demonstrate more comprehensively the matching of the two approaches. Figure 8.1(a) displays the results for the small order de Bruijn network ($M_1 = 2,048$) while Figure 8.1(b) displays the results for the larger de Bruijn network ($M_2 = 16,384$). Note that the vertical y -axis is logarithmically scaled. One can observe a very close fitting of the simulation results to the analytical approximation. It is only the last hop that differs slightly. This deviation can be explained by the following two facts. On the one hand, the theoretical values are merely an approximation of the real values. On the other hand, the simulation environment encounters factors not present in analytical methods, e.g., the network topology may temporarily differ from the targeted structure.

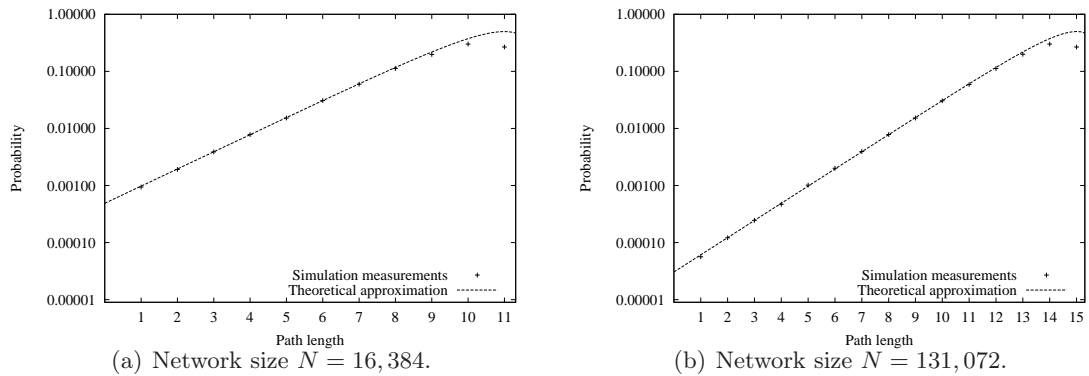


Figure 8.1.: Shortest path distribution.

It can be observed that de Bruijn graphs expand exponentially. The majority of the shortest paths between pair of nodes is very close to the diameter of the network. This explains the results of the following section on the average query length.

8.2.2. Impact on Average Query Length

The average query length can be regarded as the most relevant metric to evaluate the scalability of a P2P overlay network. Moore's law defines lower bounds for k -regular graphs as it is provided in Inequality 2.3. In fact, de Bruijn graphs are asymptotically optimal graphs converging to this bound for sufficiently large graph order N and node degree k . As it is shown in [LKR03]:

$$\mu_{DB} \approx D_{DB} - \frac{1}{k-1}, \quad (8.2)$$

where μ_{DB} is the average asymptotic distance in de Bruijn graphs and D_{DB} is the diameter. Since the diameter of the network is already close to the Moore bound (cf. Inequality 2.2), the average distance cannot shrink much further.

The relevant collected measurements include experiments involving both the Omicron and the Chord networks. Initially, we consider only the scalability of the Omicron network with varying average cluster size between three values ($\bar{K}_1 = 4$, $\bar{K}_2 = 8$, $\bar{K}_3 = 16$). Figure 8.2 provides the graphical representation of the experiments. Both x -axis and y -axis are logarithmically scaled to provide a more comprehensive view. The average query length is constituted both of the inter-cluster routing (de Bruijn based) towards the targeted cluster and the intra-cluster step to reach an Indexer. In the particular experiments routing is performed by peers assigned only with the Router role. No Cachers are present in this experiment, therefore, Routers access directly an Indexer of the cluster to get the queried information. The close matching of the analytical approximation and the simulation results can be evidently observed from this figure. Moreover, it can be stated that, as the size of clusters grow exponentially, the average routing length decreases linearly. Therefore, as the cluster maintenance cost grows exponentially, it can be safely assumed that there is an optimal maximum value for the size of the clusters, beyond which the cost grows more quickly than the utility.

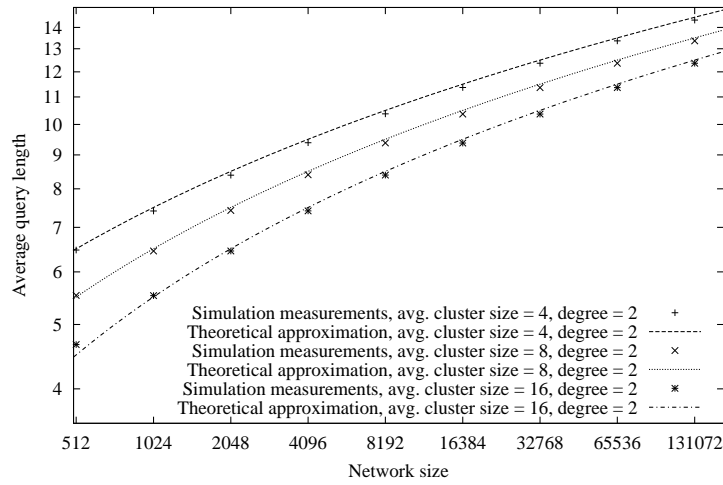


Figure 8.2.: Routing scalability.

Beyond the absolute performance of Omicron, it is essential to compare it with a widely known system like Chord, which can be considered as a reference point. Figure 8.3 shows the related simulation results. It should be noted that both X and Y axes are logarithmically scaled to provide a more comprehensive view. As it can be observed from this figure, Chord outperforms the Omicron network configuration with average cluster size $\bar{K} = 8$ and node degree $k_1 = 2$. However, it should be noted that Chord's design requires a logarithmically increasing node degree. Though, by similarly increasing the node degree of Omicron, the average query length is getting significantly smaller than Chord. To analytically express this, if the diameter in Equation 8.2 is replaced by the formula of Equation 5.1 we get:

$$\mu_{DB} \approx \frac{\log(N)}{\log(\log(N))} - \frac{1}{\log(N) - 1}. \quad (8.3)$$

Chord's average length approximation is given by $\mu_{CH} = \log(N)/2$. Therefore, for similar node degree, Omicron achieves smaller average query length. In fact, for the network order range described in Figure 8.3, by setting Omicron's node degree to $k_2 = 4$, Omicron outperforms Chord. Again here, it can be noted the good matching of the analytical approximation and the simulation results.

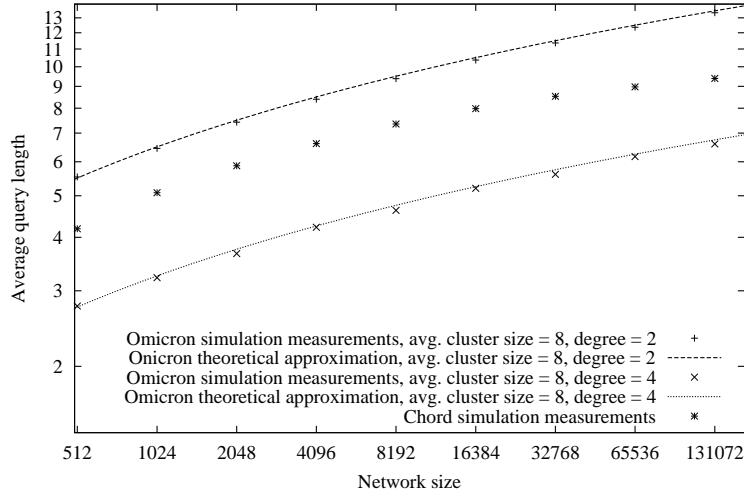


Figure 8.3.: Routing scalability: Omicron versus Chord.

8.3. Network Proximity

The focus of the previous section is on the average shortest paths between nodes. The length of the path is expressed in number of hops required to reach the final destination. Each step contributes equally with weight 1. While such a metric expresses well the complexity of the designed network, it lacks the ability to capture the accomplished efficiency on mapping the P2P overlay network to the underlying network infrastructure.

In fact, many P2P overlay networks do not consider at all the efficient mapping in their design. For instance, Chord's algorithm determines uniquely the neighbor peers (both successors and fingers) of each individual peer. Therefore, peers cannot select neighbors that are closest.

In contrast, Omicron has this freedom since any peer with the required role in a cluster is similarly adequate⁴⁶. It is the purpose of this section to explore this ability quantitatively and observe the improvement in end-to-end distance.

In order to define the underlying network distance between two peers, the two-dimensional torus model with Euclidian distances, which has been introduced in Section 7.3, is utilized. The cost of a connection is analogous to their distance, which we shall call "*virtual delay*". For the performed experiments, the Euclidian space contains a 2×2 area. Therefore, the maximum virtual delay between two peers is $\sqrt{2}$ delay units (remark the properties of toruses). Also, it should be noted that the calculated distance ignores the random effects introduced in the simulator to estimate the transmission delay.

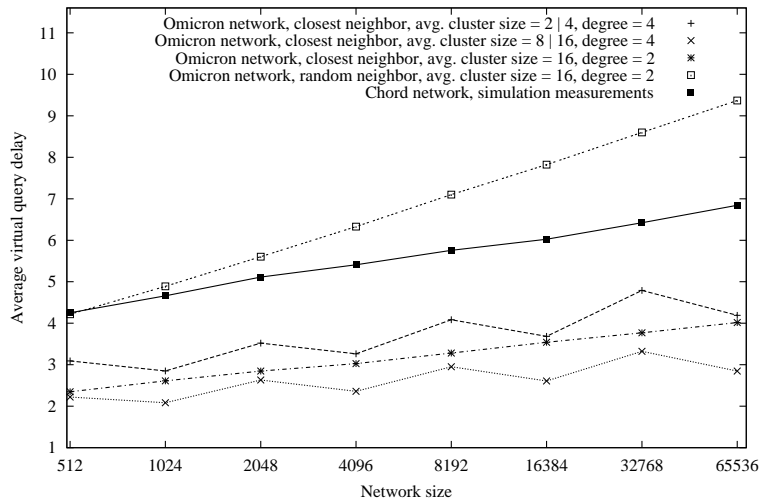


Figure 8.4.: Vicinity exploration: Omicron versus Chord.

The performed experiments include four different Omicron configuration settings and a typical Chord network. The average cluster size varies between 2 to 16 peers and the inter-cluster node degree varies between 2 and 4. Two different policies are utilized. The first one is a *random neighbor selection* based policy, unaware of the distance between the peers. It has similar effects to the policy of Chord. The second policy selects the closest one among the peers of the neighbor cluster. However, in the performed experiments peers may select only their neighbor Routers. Their connection to the local Indexer (belonging to the same cluster) is locality unaware. The last selection has been introduced to reflect scenarios where a small number of reliable peers assigned with the Indexer role are available. This "greedy" algorithm is expected to perform more efficiently than the random selection policy.

⁴⁶We ignore issues such as trust or reliability at this selection. However, they are important aspects for consideration.

As it can be observed from Figure 8.4, the Omicron network based on the random selection policy and the Chord network perform inferiorly compared to the closest neighbor approach. In fact, their relative performance is analogous to the hop based distance metric (studied in the previous section) since the large number of uniformly distributed peers is the determining factor. In contrast, the closest neighbor based experiments exploit peer vicinity much more efficiently. In fact, the performance is even better when peers (Routers) can optimally select their Indexer, too. Further, the non-monotone curves observed in cases where the inter-cluster node degree is 4 is an interesting point. The explanation for this effect is based on the fact that some network sizes cannot be defined as integral powers of the node degree (4). Therefore, the average cluster size varies between two values (e.g., 2 and 4 or 8 and 16).

8.4. Routing Workload Distribution

Omicron has been specifically designed to deal efficiently with the heterogeneity of the peers. In order to do this, several mechanisms have been introduced. However, it is desirable to accomplish a relatively even distribution of the workload among peers assigned with similar roles. This section focuses specifically to the distribution of the routing workload, aiming to find potential bottlenecks that can degrade the performance of the system.

8.4.1. Query Workload Distribution

In Section 8.3, we compared the accomplished efficiency in mapping the overlay connections to the underlying network cost. It has been observed that vicinity aware neighbor selection can considerably improve the observed end-to-end delay. However, it is interesting to examine how the described greedy algorithm affects the routing workload distribution among the Routers.

On the one hand, Figure 8.5(a) provides the query workload distribution for a typical Omicron configuration, where neighbors are randomly selected. The order of the network is $N = 1,024$ peers, where the inter-cluster node degree is $k = 2$ and the average cluster size is $\bar{K} = 16$. A relatively even workload distribution can be observed. On the other hand, Figure 8.5(b) shows the accomplished workload distribution when peers select their neighbor based merely on proximity criteria. Apart from the connection policy, the two networks are identical.

Obviously, the greedy algorithm has a negative effect on the load balance. For certain pairs of peers the assigned relative workload ratio is as high as 50 (regarding the specific network configuration described above). Exploring more deeply the details of the simulation experiments the critical factor has been identified, which results to this undesirable effect.

For single peer nodes of de Bruijn networks, it is guaranteed that both the incoming and the outgoing node degrees are equal and fixed (with the exception of particular nodes,

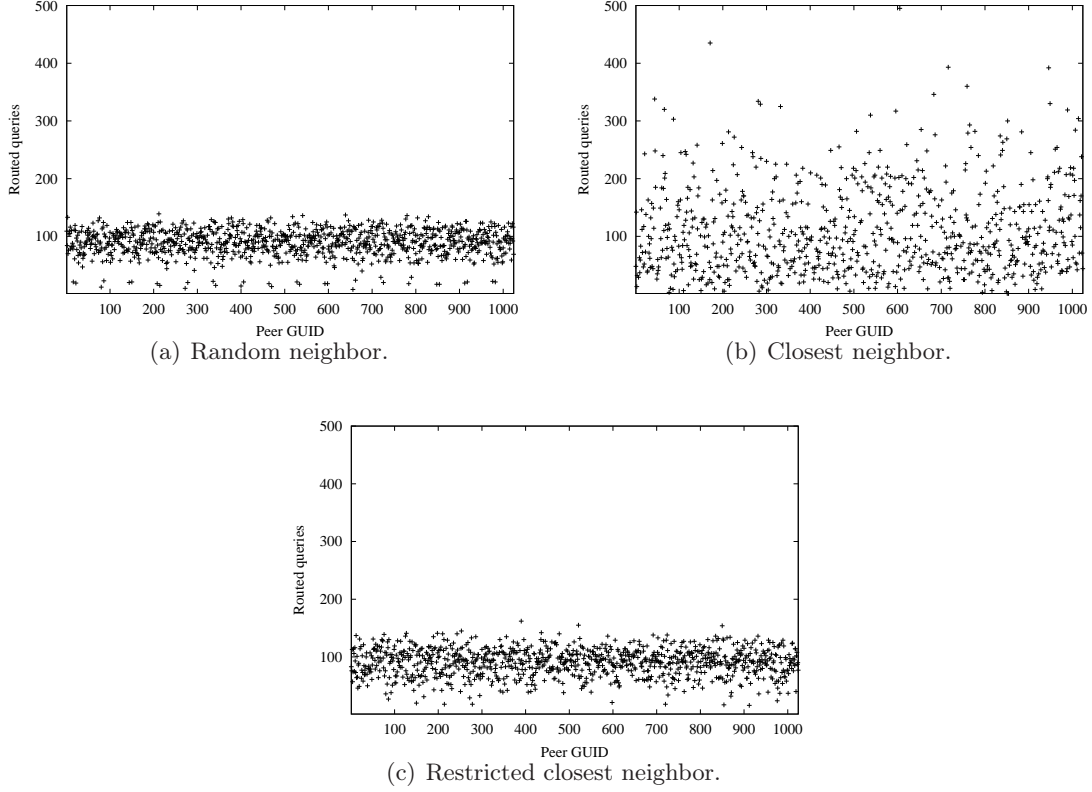


Figure 8.5.: Query routing load distribution.

as it is discussed in Section 5.3. However, for cluster based de Bruijn networks there is no such guarantee. In fact, only the outgoing degree is equal and fixed ($k = 2$ for the investigated scenario). In contrast, the random neighbor based network is constructed by enforcing also the equal and fixed incoming node degree, resulting in the observed load distribution.

Therefore, it is rational to examine whether by applying this restriction to the vicinity aware network (*restricted closest neighbor*), improved load distribution results could be achieved. However, such a restriction results to non optimally selected neighbors. Figure 8.5(c) shows the results of such a scenario. Peers accept new neighbors as long as their incoming degree is less than or equal to their outgoing degree. If a new connection request arrives while they have reached their maximum allowed incoming degree, the connection request is refused and the originator peer requests a connection from a different peer. It should be noted that such a policy is not optimal. A better strategy can be exploited, e.g., by disconnecting the furthest among the currently connected peers. Nevertheless, even this simple policy can provide the desirable results as we can see in Figure 8.5(c). Practically, the query routing load distribution is indistinguishable between randomly selected neighbors and the restricted closest neighbor selection policy. Further results for much larger network sizes are provided in Section D.

8.4.2. Workload Distribution and Efficiency

The experiments performed in the previous section evaluating the routing workload distribution revealed that the restricted closest neighbor policy has a desirable distribution. However, it is not expected to exploit the network vicinity as efficiently as the greedy algorithm. Therefore, it is essential to evaluate the ability of the new algorithm to exploit network vicinity.

Further experiments have been performed (similar to the ones in Section 8.3). The network size varies between 512 to 65,536 peers. Peers submit queries to randomly selected destinations. The queries are forwarded to the targeted cluster via Routers belonging to neighbor clusters; then, a local Indexer (of the targeted cluster) replies.

Figure 8.6 provides the collected results where the x -axis scales logarithmically in order to provide a more comprehensive view. The three different policies are graphically displayed with curves. As it can be observed, the restricted closest neighbor policy performance is close to the greedy closest neighbor policy. It is expected that a more advanced policy that disconnects the furthest among the connected neighbors can provide even better results. Moreover, if the random neighbor Indexer policy is replaced by a vicinity based one, both vicinity aware policies perform better.

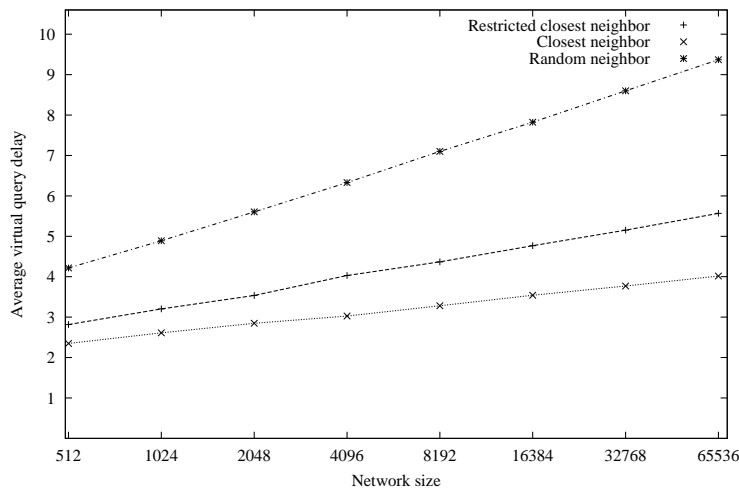


Figure 8.6.: Vicinity exploration versus load-balance.

8.4.3. Enhanced de Bruijn Topology

As it has been mentioned in Section 4.5.4, particular de Bruijn nodes that are self-connected receive a significantly less routing workload compared to the rest of the nodes. The provided solution develops an enhanced de Bruijn structure where self connected links are removed and replace by others that connect these nodes (c.f. Figure 4.7). The focus of this section is to evaluate the improvement accomplished with the new structure.

8. Evaluation Experiments

The related experiments involve the two different de Bruijn digraphs. During this experiment, one peer per cluster (a Maintainer) submit queries targeted to each existing cluster of the network (resulting in $M^2 - M$ queries, where M is the number of clusters). The collected measurements include the number of messages forwarded by each cluster (aggregating all the forwarded messages of Routers belonging to each particular cluster). The node degree is $k = 2$ and the results shown in Figure 8.7 involve 512 constructed clusters. The y -axis scales logarithmically in order to provide a more comprehensive view.

As it can be observed from Figure 8.7(a), self connected clusters having GUID (000000000) and (111111111) receive 511 ($M - 1$) queries. This means that only the queries that are targeted to these clusters arrive there and no further queries are forwarded to other clusters (the initial submission of the query is not accounted in these measurements). Certainly, this is not a desirable behavior.

In contrast, by examining Figure 8.7(b) we note that the workload of the modified clusters (by replacing their self-connected links) is considerable higher (twice the load observed in the other scenario). It should be noted that Algorithm 5.3.3 used in this experiment aims mostly at providing optimal shortest paths. By replacing this algorithm it is possible to receive even better routing workload balance at the cost of longer paths. Such an algorithm is described in [SR94].

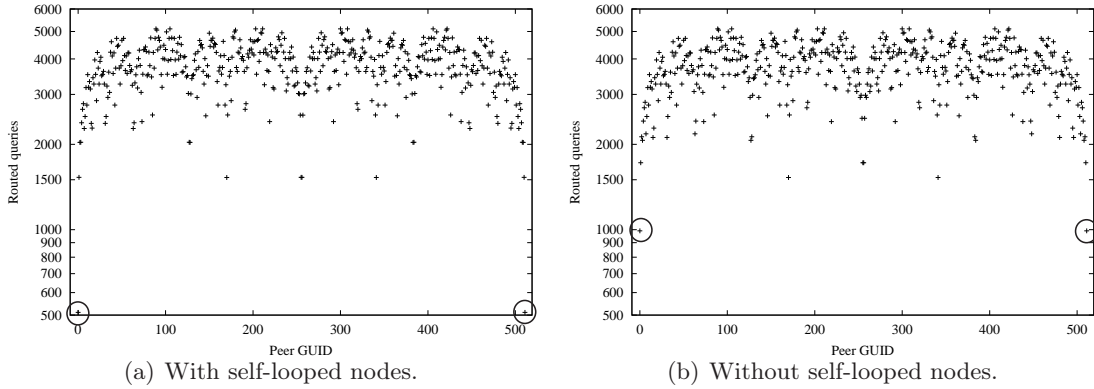


Figure 8.7.: Routing query distribution on de Bruijn graphs.

Aiming to explore the effect of the enhanced de Bruijn topology with a wider network size range, we provide a comparison of the routing workload distribution between the two topologies. Figure 8.8 shows the achieved load on the self-connected clusters where both X and Y axes scale logarithmically in order to provide a more comprehensive view. As it can be observed the enhanced topology distributes almost 50% additional load to the self-connected clusters compared to the original structure for all ranges of the explored network sizes. It should be noted that the x -axis is the size of the de Bruijn network and not the size of the complete Omicron network.

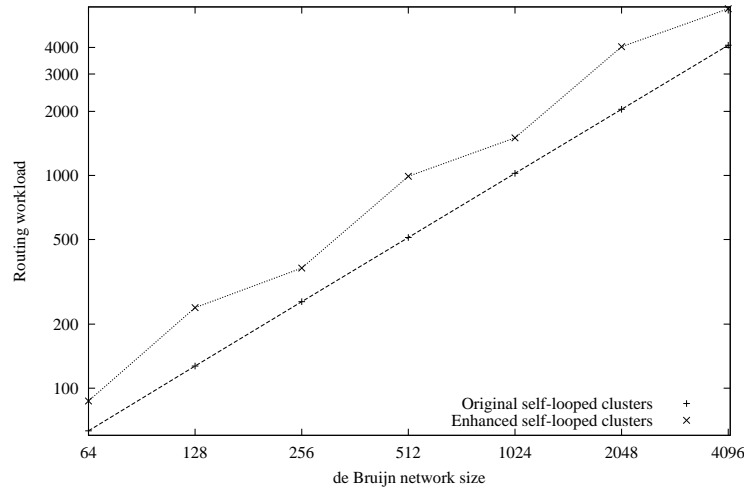


Figure 8.8.: Routing workload on self-looped nodes.

8.5. Maintenance Overhead and Accomplished Cluster Coverage

When new peers request to join an Omicron-based P2P system, Maintainers perform a number of operations in order to place the new peers in the network. The purpose of their actions is to achieve a well balanced topology where clusters have sufficient endurance and the total workload is minimized and well distributed.

Obviously, the best selection can be made when the endurance of each cluster of the network is known. However, such a solution is very costly as the scale of the system increases considerably. Therefore, as it has been mentioned in Section 4.5.1, Maintainers perform a random walk collecting the endurance of each cluster in order to decide which one is the best selection to direct the new peer to.

In this section, we evaluate the ability of four different algorithms to accomplish an effective cluster coverage at low cost. Three of them are probabilistic and one is deterministic. All algorithms start randomly at any peer, assuming that new peers randomly select their first contact to send the request to. Even if the employed bootstrap phase does not comply with this assumption, it can be easily achieved by performing an additional random walk before the sampling phase begins.

8.5.1. Investigated Algorithms

Probabilistic Algorithms

The probabilistic algorithms differ both in length and neighbor selection policy. Their description is provided in the following list.

1. **Random destination.** This algorithm starts from any random peer, which randomly selects the final destination. This has the advantage of simplicity since it does not differ from a typical query routing procedure. However, the number of covered clusters is equal to the average query length evaluated in Section 8.2. Therefore, the average random walk length is given by Equation 8.2, which is shorter than the network diameter. The achieved cluster coverage is very similar to the assigned routing workload evaluated in Section 8.4.3.
2. **Short random walk.** This algorithm starts from any random peer by randomly selecting only the next peer to follow among the neighbors found in the routing table. The procedure is recursively applied until a random walk of length equal to the diameter D of the network is reached. This algorithm has the advantages of equal length random walks and better coverage distribution than the previous algorithm since seldom reached clusters are more likely to be visited. However, its implementation is more complex. It requires a non-oblivious routing mechanism to avoid cycles in the random walk. Clusters should be visited only once.
3. **Long random walk.** This algorithm is very similar to the previous one. The only difference is that the required length of the random walk must be twice the length of the diameter ($2 \cdot D$). It is expected that the longer random walk combined with the cycle avoidance restriction will provide a much better cluster coverage. The disadvantage of this algorithm is that it costs twice as much as the short random walk.

Deterministic R-Shift Algorithm

The aforementioned algorithms perform random walks in order to sample the endurance of the clusters. In this section, a deterministic algorithm is investigated in order to evaluate such an alternative. It is assumed that the deterministic walk begins randomly at any peer (as also assumed in the previous cases).

There are certain restrictions and guidelines in designing an effective deterministic walk appropriate to sample clusters' endurance.

1. The length of each walk must be as close as possible to its maximum value (the diameter of the network).
2. The length of each walk should not differ considerably (independently of the position of the initial cluster).
3. The cluster coverage should be as wide and as evenly distributed as possible.

There are several algorithms that can fulfill the aforementioned requirements. We have designed one that is as simple as possible. It is called "*R-Shift*" algorithm. Basically, each peer deterministically select the final destination by applying a *right-shift* operation at the GUID of its cluster (note that the conventional routing in Omicron utilizes *left-shift* operations). The new symbol at the right end of the GUID must be different than the symbol at the left end before the right-shift operation. Formally, this operation can be

expressed as follows.

$$rshift(u_1u_2...u_D) = u_2...u_D\overline{u_1}, \quad (8.4)$$

where the $\overline{u_1}$ is an operation that provides a different symbol from the available alphabet (deterministically). For example, for binary de Bruijn graphs, it holds that $\overline{0} = 1$ and $\overline{1} = 0$. It is guaranteed that using the R-Shift algorithm all the clusters will be included in the sampling since Equation 8.4 provides a direct and unique mapping of the input cluster to the output cluster.

This algorithm is graphically illustrated in Figure 8.9, which displays a de Bruijn(2,4) digraph. Two different deterministic walks are shown. Assume that the two walks start at nodes (0011) and (0101), respectively. Thereby, the R-Shift algorithm produces the sequence (0011) \rightarrow (0110) \rightarrow (1100) \rightarrow (1000) \rightarrow (0001) for the first case, which is traversed by **Msg1**. Similarly, the sequence (0101) \rightarrow (1010) \rightarrow (0100) \rightarrow (1001) \rightarrow (0010) is traversed by **Msg2**. However, in certain cases the length of the path is smaller than the diameter, e.g., (1101) \rightarrow (1011) \rightarrow (0110).

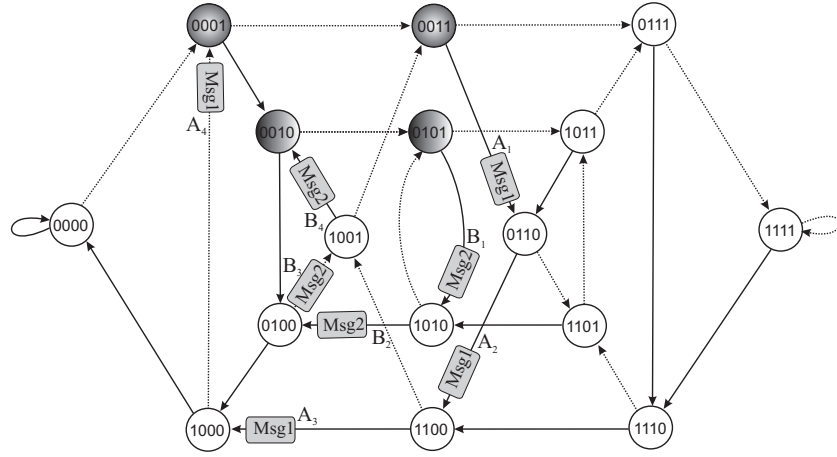


Figure 8.9.: Deterministic R-Shift algorithm.

The described algorithms have different inter-cluster communication cost that determines the sampling walk length. Table 8.1 summarizes this cost.

Table 8.1.: Sampling algorithms routing cost.

Sampling algorithm	Expected walk length
Random destination	$D_{DB} - \frac{1}{k-1}$
Short random walk	D_{DB}
Long random walk	$2 \cdot D_{DB}$
R-Shift	$\approx D_{DB}$

8.5.2. Cluster Coverage

The evaluation of the four sampling techniques is presented next. The most critical aspect we encounter is their ability to visit evenly each cluster of the network. Figure 8.10 provides the results for a specific Omicron configuration where the structured de Bruijn network is composed of 2048 clusters and the inter-cluster degree is $k = 2$. Figure 8.10(a) describes the coverage distribution (the number of times each cluster is sampled) for the random destination algorithm. Similar results are provided in Figure 8.10(b), Figure 8.10(c) and Figure 8.10(d) for the R-Shift algorithm, the short random walk algorithm and the long random walk algorithm, respectively.

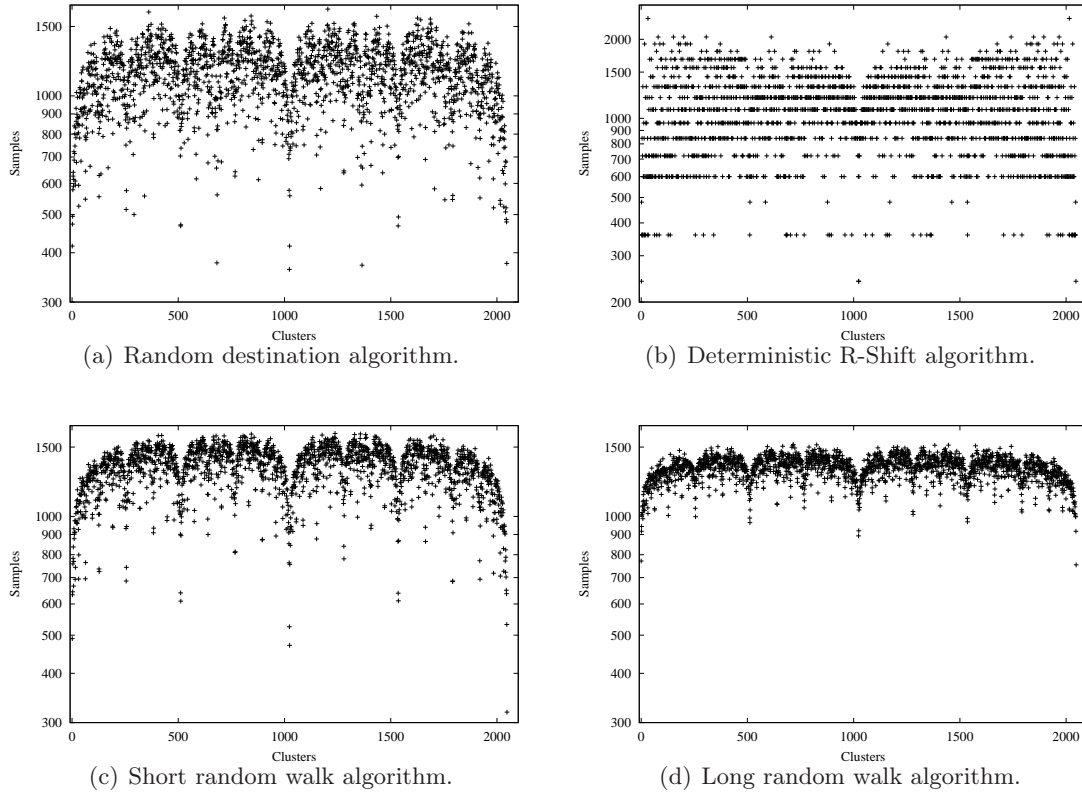


Figure 8.10.: Sampling distribution using random walks.

As expected, the long random walk algorithm provides the most evenly distributed sampling where the majority of samples differ less than 10% from the mean value. The reason for this lies on the fact that more clusters are sampled at every join, which is combined with the cycle-removal mechanism (non-oblivious routing mechanism). Therefore, clusters that are seldom sampled by other algorithms have a higher probability of sampling by this algorithm.

Aiming at providing additional results on the ability of each algorithm to evenly sample the network clusters, further experiments have been performed. In these experiments the size of the de Bruijn network is modified between 64 and 16,384 clusters. For each experiment,

a number of join requests is generated that is related to the size of the network and the utilized algorithm. The target is to generate approximately equal workload for all of the algorithms. The *mean value* of samples per cluster are provided in Table 8.2.

Table 8.2.: Mean value of cluster coverage sampling.

de Bruijn network size	Mean value of samples
64	716.554
128	834.781
256	957.398
512	1,078.584
1,024	1,199.581
2,048	1,319.604
4,096	1,439.903
8,192	1,559.918
16,384	1,679.971

The quantity of interest in these experiments is the evaluation of the *standard deviation* of the cluster sampling distribution for each algorithm. Figure 8.11 summarizes the results of the experiments. It should be noted that the x -axis scales logarithmically in order to provide a more comprehensive view. As it can be observed, the long random walk algorithm achieves the smallest standard deviation for the complete range of the evaluated network sizes. Moreover, the short walk algorithm achieves better performance compared to the random destination algorithm as the network grows. The standard deviation of the R-Shift algorithm grows considerably more compared to the three probabilistic alternatives.

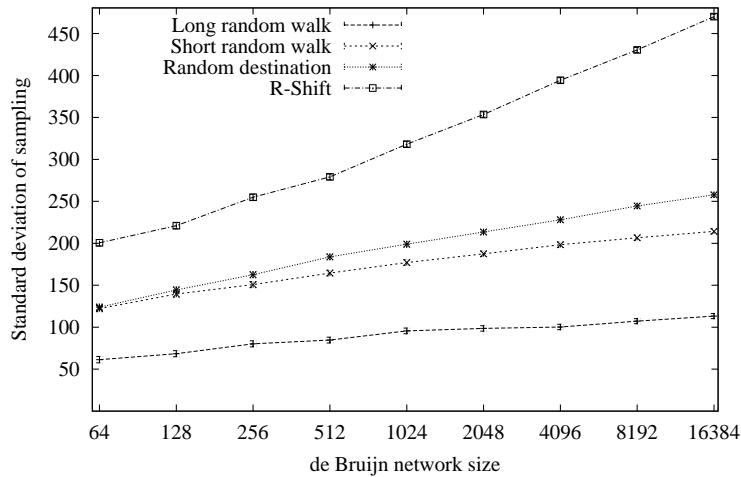


Figure 8.11.: Standard deviation of sampling in de Bruijn networks.

However, the standard deviation provides an absolute value for the effect. In many cases, it is more important to observe a relative metric that relates the standard deviation with the

mean value. Such a metric is the *coefficient of variation*, which is defined as $CV = \sigma/\mu$, where σ is the standard deviation and μ is the mean value. Therefore, by combining the mean values provided in Table 8.2 and the standard deviation from Figure 8.11 we can get the coefficient of variation for the experiments. It should be noted that the mean value of the deterministic algorithm differed from the provided mean values set. Therefore, it is not included in the provided report. Figure 8.12 summarizes the results on the coefficient of variation. As it can be observed, the long random walk algorithm accomplishes always a coefficient of variation less than 10%. Also, it is interesting to notice the decreasing rate of CV for the short random walk algorithm. It can be stated that for a very large network size, its performance approaches asymptotically the performance of the long random walk algorithm.

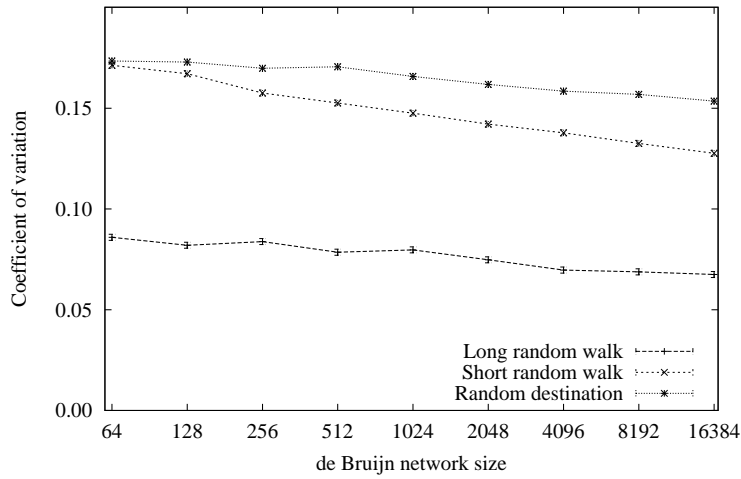


Figure 8.12.: Coefficient of variation of the cluster sampling mechanisms.

8.5.3. Cluster Sampling Inter-arrival Distribution

The aggregated behavior of the cluster sampling algorithms can be observed with the experiments performed in the previous sections. However, it is essential to evaluate how often each particular cluster is revisited in subsequent random walks.

As it has been observed in Section 8.4.3, self-connected clusters are the least frequently used clusters with respect to their involvement in the routing procedure. Similar results can be obtained from Figure 8.10. Also, by examining the details of the collected results, it has been noticed that clusters with GUID, e.g., (011001001) or (0110010011) or (01100100110) are among the most frequently visited clusters for each sampling algorithm (GUIDs with lack of "patterns" in their digit sequence).

Let us define *each random walk experiment* as a "round". The *event* of interest E_C is "*how many random walks (rounds) are necessary until a particular cluster C is sampled*". Such a quantity can provide vital information on whether the sampling algorithm is adequate for its need.

Therefore, further experiments have been performed aiming to evaluate E_C , where the GUID of cluster C is taking the values provided in Table 8.3. Collecting the experiment results for these clusters, the diagrams of Figure 8.13 have been drawn to show the probability that the particular cluster will be sampled after a certain number of sampling walks. In order to generate these measurements, the long random walk algorithm has been employed.

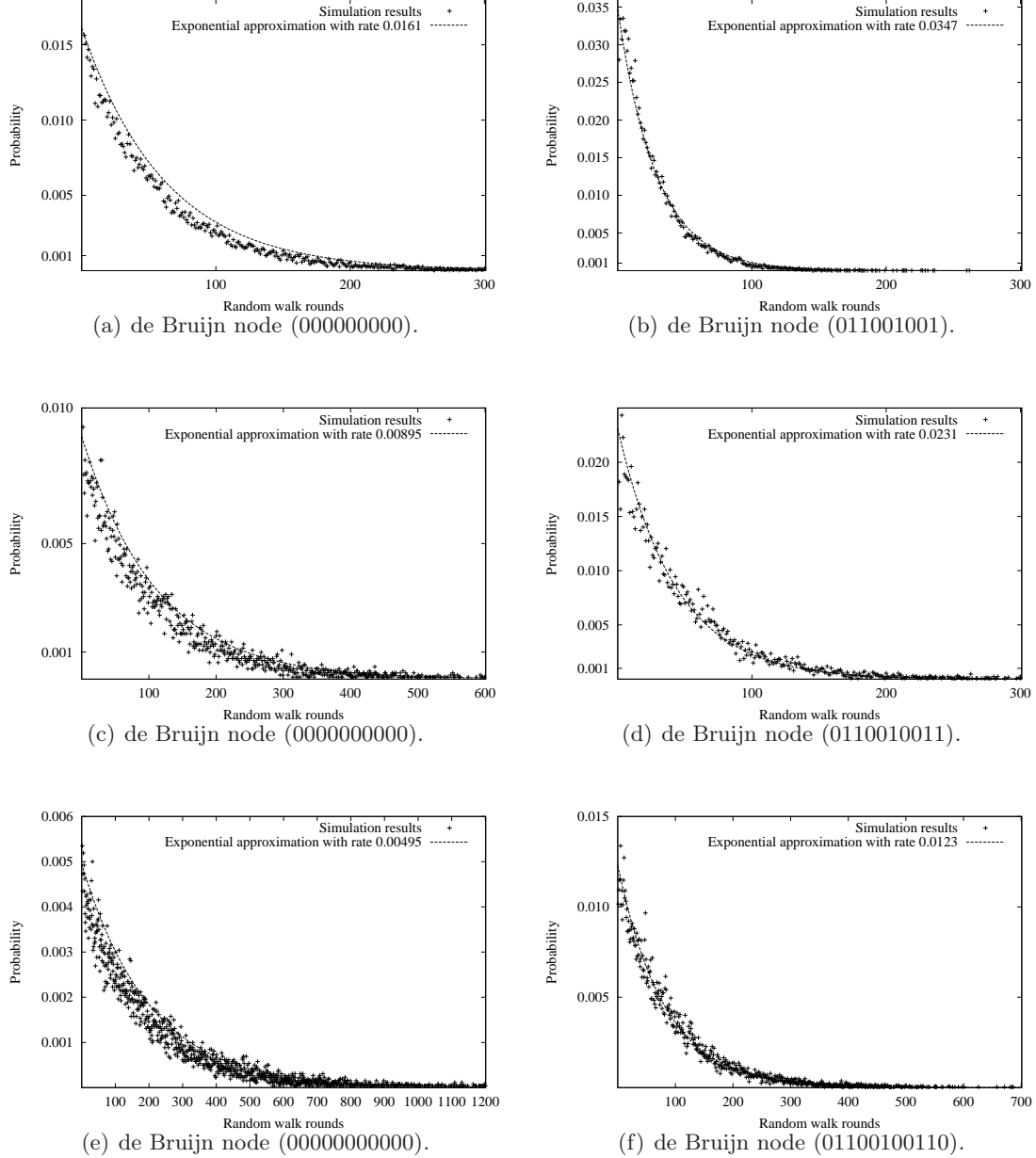


Figure 8.13.: Cluster sampling inter-arrival distribution.

It is interesting to observe that the cluster sampling inter-arrival distribution can be closely

approximated by an *exponential* distribution of the form $f(x) = \lambda x^{-\lambda x}$, $x \geq 0$. The reason for such behavior can be explained as follows. Let us call V_C the event of interest, which is visiting a particular cluster C during a random walk. V_C is a *Bernoulli* random variable:

$$g(x) = \begin{cases} g(0) = Pr\{V_C = 0\} = 1 - p, \\ g(1) = Pr\{V_C = 1\} = p, \end{cases} \quad (8.5)$$

where p is the probability of success that depends on the cluster position in the digraph. Each random walk is an independent event. If we let X be the number of the performed events until a success occurs, then X is said to be a *geometric* random variable with parameter p . Its PMF is given by:

$$h(n) = Pr\{X = n\} = (1 - p)^{n-1}p, \quad n = 1, 2, \dots \quad (8.6)$$

The geometric distribution is the discrete equivalent of the exponential distribution. Therefore, the cluster sampling inter-arrival distribution can be approximated well with the exponential distribution.

Table 8.3 provides the approximated rates of the exponential functions for the examined clusters. As it can be seen from it, seldom visited clusters have a lower rate than frequently visited clusters. Also, as the size of the network gets larger, the approximated rates are getting smaller, which is expected since more clusters are available for sampling. However, the peer join rate is getting higher, providing the necessary lower bound for the sampling rate.

Table 8.3.: Exponential cluster sampling inter-arrival rate approximation.

GUID	Rate (λ)
(000000000)	0.0161
(011001001)	0.0347
(0000000000)	0.00895
(0110010011)	0.0231
(00000000000)	0.00495
(01100100110)	0.0123

8.6. Indices Caching Mechanism Performance

In this section, the proposed caching mechanism described in Section 5.6 is evaluated with simulation experiments. In order to demonstrate the general applicability of the proposed mechanism, we perform the experiments using the Chord network instead of Omicron. The results are very similar in both of them. The goal is to evaluate the performance improvement with respect to the routing workload by comparing the proposed indices cache mechanism with the original network.

The population of the simulation network consists of 4,096 peers distributed randomly over a Chord ring with key range of 65,536 values. Peers and resources share the same

key range. Each experiment lasts approximately 30 minutes of simulation time. Peers randomly select a resource to query every 20 seconds (asynchronously from each other). The process is repeated 80 times resulting in a total number of approximately 327,000 queries.

Peers start requesting the resources after a certain stabilization period. The probability distribution of the resource selection follows a lognormal distribution with parameters $\mu = 0.82$ and $\sigma = 2.9$ following the guidelines in [BBR04]. The selection of the lognormal distribution over the Zipfian distribution is motivated by the greater challenge of the former since the popularity of the resources is more widely distributed. The implemented lognormal generator produces randomly selected GUIDs limited to the aforementioned key range. On average, approximately 4,000 – 4,100 different keys are generated on each run.

Figure 8.14 displays a representative CDF of the resource popularity, where the resources are sorted from the most popular to the least popular. From this figure, it can be concluded that the first 25 most popular resources contribute to approximately 80% of the query load. Thus, an equivalently small cache size is adequate to store them and achieve a high performance, provided that the popularity identification algorithm operates correctly. Nevertheless, in real experiments, the cache size may have to be bigger to capture effectively the popular resources since the key range may be considerably larger.

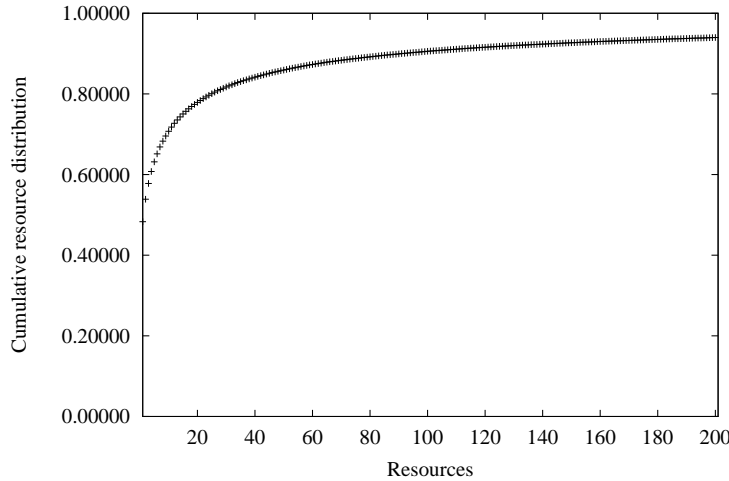


Figure 8.14.: Cumulative resource popularity distribution.

Figure 8.15 displays the reduced routing communication cost in terms of required hops, as the percentage of the communication cost of the original Chord network. The communication cost is evaluated as a function of the cache entry expiration timeout. Two different experiments have been selected:

1. Experiment A, where the *FREQUENCY_THRESHOLD* is 5, the maximum cache size is set to 80 and the frequency counter is reset every 200 seconds.

2. Experiment B, where the *FREQUENCY_THRESHOLD* is 3, the maximum cache size is set to 300 and the frequency counter is reset every 100 seconds.

We can observe that the total communication load for query routing can be considerably reduced using the caching mechanism down to 50% of the original load.

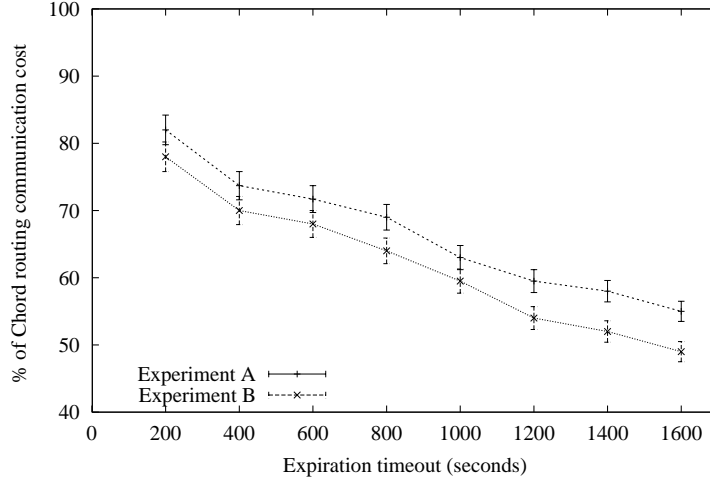


Figure 8.15.: Cache routing load as a percentage of the original Chord network.

Moreover, peers responsible for popular resources may become "hot spots" and potential bottlenecks of the system. By utilizing the cache mechanism the load for replying to the queries is getting more evenly distributed. Figure 8.16(a) displays the load balance in the original Chord network, while Figure 8.16(b) shows the query replying in the cache-enhanced Chord network. It should be noted that the vertical y -axis is logarithmically scaled. Moreover, many peers reply with cached values which are not considered in this figure. Otherwise, since the number of replies is increased because the reply should be provided first to the intermediate peer and then to the originator of the query, it would have been misleading.

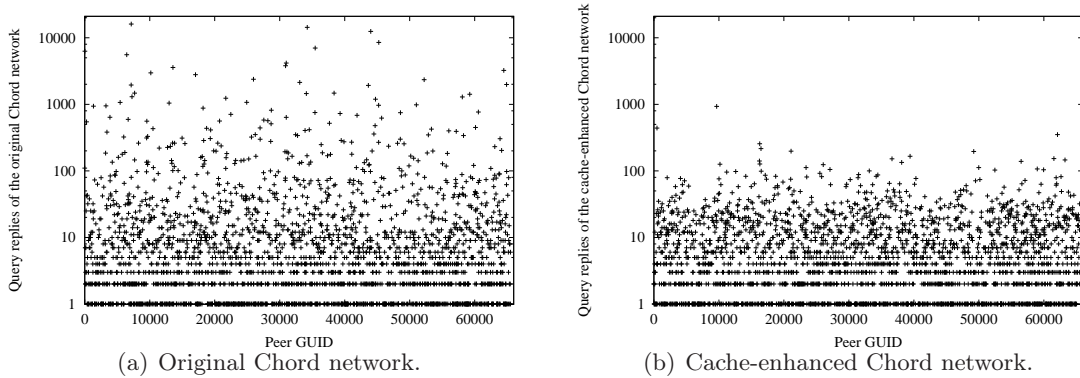


Figure 8.16.: Load distribution for replying queries.

8.7. Summary

This chapter has provided the quantitative results of the experiments performed on Omicron and Chord in order to evaluate their performance in several key scenarios. Mostly, the focus has been put on aspects involving inter-cluster communication mechanisms, since these scenarios have the largest complexity. The performed experiments vary in network sizes and include network topologies composed of more than 130,000 peers, demonstrating the performance abilities of the developed simulation framework.

The ability of Omicron to meet several critical non-functional requirements for P2P overlay networks has been evaluated by simulation. The *scalability* of the routing process follows accurately the analytical approximations provided in the literature, both for the average and the worst case scenarios. Similar results are provided for the distribution of the shortest paths between peers. Further, the ability of Omicron to efficiently exploit the *vicinity* in the underlying network topology has been shown. Multiple algorithms have been investigated to additionally provide an *evenly-distributed* routing workload. The intrinsic load balance-related limitations of de Bruijn digraphs have been evaluated together with the suggested *enhanced* digraph topology.

Moreover, the *maintenance overhead* that is required to ensure network *stability* through cluster *endurance* has been evaluated with multiple sampling algorithms. Both deterministic and probabilistic algorithms have been employed where the latter showed better cluster coverage capabilities. In addition, the cluster sampling inter-arrival distributions have been estimated revealing an interesting distribution property that can be further exploited by analytical means to obtain a stochastically described P2P network. Finally, the usefulness of caching mechanisms for non-uniformly distributed queries has been evaluated by applying on Chord the suggested mechanism found in Section 5.6. Thus, both the general applicability of the mechanism is illustrated and the improved routing performance is shown.

Conclusions and Outlook

Ithaka gave you the marvelous journey. Without her you wouldn't have set out. She has nothing left to give you now. And if you find her poor, Ithaka won't have fooled you. Wise as you will have become, so full of experience, you'll have understood by then what these Ithakas mean.

K. KAVAFIS, ITHAKA

Conceptual outline.

The performance evaluation provided in Chapter 8 summarizes the results gathered during the simulation-based experimentation phase. Thereby, it offers a necessary proofs to justify the design decisions taken in Chapter 4 and quantitatively evaluate the algorithms and mechanisms described in Chapter 5. In this chapter, we summarize the crucial aspects of the proposed overlay network design and the key mechanisms. Further, we discuss the concluding remarks of this work and its importance for the research community. In addition, the described design opens new directions for further research. Thereby, some appealing topics are presented aiming to trigger the interest for deeper investigation in P2P overlay network mechanisms.

This chapter is organized as follows. Section 9.1 summarizes the crucial issues introduced in this thesis. Afterwards, Section 9.2 concludes the contribution of this dissertation. Finally, Section 9.3 provides an outlook of this work aiming to trigger interesting research topics for further study.

9.1. Summary

This thesis addresses the architectural design of overlay networks to flexibly support the communication needs of P2P systems. Its aim is to address a large set of crucial requirements, supplied with advanced algorithms and mechanisms. This is a complex procedure especially when it targets large scale, highly dynamic and heterogeneous environments. In such scenarios many trade-offs have to be considered between the large variety of distinct requirements and characteristics. The research presented in this thesis has addressed the most important of these requirements. Moreover, it has identified the related trade-offs and conflicts as well as the limitations of the current approaches.

More specifically, a systematic identification of requirements relevant to P2P overlay networks, such as scalability, evenly distributed workload and support for heterogeneity is provided in Chapter 2. The analysis of representative P2P application types is utilized in this investigation in order to discover comprehensively the relevant requirements. Further, a plethora of related design approaches for P2P overlay networks is given in Chapter 3. Particular emphasis is put on structured and hybrid solutions.

To solve these issues, a novel architecture, called Omicron, has been developed that provides concurrently *scalability, incremental expandability, fault-tolerance, stability, load-balance and efficiency*. In addition, the proposed system addresses the effective collaboration of peers with heterogeneous capabilities and behavior. This architecture is described in Chapter 4.

Several key mechanisms are involved in the proposed overlay network architecture: (i) an efficient overlay structure based on de Bruijn digraphs; (ii) a clustering mechanism that partitions the peers into stable groups that form the components of the overlay; (iii) a role adaptation scheme inside the clusters that effectively matches peers' capabilities and behavior with their responsibilities; (iv) a priority and rule-based service differentiation that provides appropriate incentives for the various roles; (v) a unique combination of structure and randomization in a two-tiered topology, as it is achieved by the usage of de Bruijn graphs and the freedom to select the neighbors; (vi) a dual identification scheme that uniquely identifies each peer without any constraints and groups them under the shared cluster identifier; (vii) an efficient caching mechanism that operates without introducing any additional signaling protocol; (viii) a burn-in based mechanism that capitalizes on conditional reliability based methods to optimally assign roles to peers. The aforementioned mechanisms are described in Chapter 5 and Chapter 6.

Further, a simulation framework has been implemented to evaluate the performance of the Omicron architecture. The developed open architecture simulator is discussed in Chapter 7. In this chapter we describe the utilized software design patterns as well as the core components and the provided functionality layers of the framework. Following the simulator description, the performance of Omicron is evaluated and compared with the respective performance of Chord using the developed tool. The collected results comprehensively demonstrate the improved performance of Omicron compared to the performance achieved by Chord. The evaluation experiments and the collected results are described in Chapter 8.

9.2. Conclusions

A crucial lesson learnt in the investigation of P2P overlay networks is that they are *complex systems* with several *conflicting requirements*. Their fulfillment is a challenging process that can be better addressed with carefully designed *hybrid* solutions that provide *flexibility* and *adaptability*. *Multi-tier* architectures are able to distinguish issues *vertical* to each other and address them separately, thus, reducing the overall complication. However, a wise selection of the minimum necessary layers is mandatory to reduce the unavoidable cost of inter-layer *co-operation* and structure *complexity*.

Moreover, the well-known *divide and conquer* strategy is successfully applied in P2P overlay network design. Several mechanisms have been introduced in this thesis inspired by this strategy. The goal of the *clustering* mechanism is twofold. The inter-cluster overlay network has considerably smaller *order* than a flat network and the required *redundancy* in providing operations and services is *restricted* to a small size community that can be easier controlled. By identifying the common core *roles* and separating their functionality, it is possible to operate the system at significantly lower cost. The peer population involved in providing particular operations and services is smaller, thus, providing more *scalable* network characteristics. The specialization achieved with the role identification mechanism grants a solution to handle the intrinsic *heterogeneity* of peers.

Further, the freedom introduced in the network architecture with the clustering and multi-role mechanisms enables the *flexible* and *efficient* deployment of the overlay network over the underlying physical network. The *heterogeneous* peer behavior can be exploited in a positive way by identifying peers that are statistically more reliable. Therefore, these peers form the backbone of the overlay network and reduce the *cost* introduced by reliability unaware policies. Further, caching mechanisms are particularly helpful for *non-uniform* query distribution scenarios. A simple caching mechanism introduced in this thesis reduce the routing workload in structured P2P overlay networks without requiring out-of-band signaling protocols. The solution considers peers' *uptime* to validly maintain the cached information.

Additionally, the role-based approach is especially beneficial for the development of an open architecture simulator, which can be adaptively configured to meet the design guidelines of multiple P2P systems with different needs. The developed tool has proven this statement by providing reusable components to build Omicron, Chord, Gnutella and JXTA based environments and evaluate their performance.

The proposed architecture and the related mechanisms have been quantitatively evaluated in order to meet a large set of the identified non-functional requirements. Omicron has proven with its evaluation that it outperforms Chord. In addition, it has been designed to efficiently meet a larger set of requirements than Chord can address.

In summary, the main contributions of this thesis is the novel architecture for P2P overlay networks that can adaptively and flexibly meet several non-functional requirements. Moreover, novel mechanisms such as the introduced endurable clusters and the adaptive role assignment supply the architecture with the efficient solutions to meet the identified requirements.

9.3. Outlook

The accomplishments described in this thesis advance the state-of-the-art in developing P2P overlay networks. The presented set of mechanisms addresses the challenges described in the introductory chapter. In addition, the described design opens new directions for further research. Thereby, some exciting topics are presented in this section aiming to trigger the interest for deeper investigation in P2P overlay network mechanisms.

Throughout this thesis de Bruijn digraphs have been proposed to be used as the core topology of the constructed overlay network. As it has been already described, de Bruijn digraphs provide asymptotically optimal network characteristics achievable even when a fixed node degree is required. An interesting question is whether new graph structures are possible that hold the desirable properties of the de Bruijn digraphs, while additionally offer incremental expandability features.

The investigated burn-in based mechanisms capitalizes on conditional reliability concepts to optimally assign costly roles to peers. However, the provided analysis is based on empirical observations of widely deployed P2P file sharing applications. A challenging research issue is to investigate solutions that can accurately predict peers uptime without assuming a particular lifetime distribution.

Moreover, peers are autonomous entities with their own control mechanisms pre-configured or interactively controlled by users. Throughout the experiments performed in this thesis, peers act independently at any point in time. However, their actions are operations that are expected by typical users. A concern that is worthwhile to be investigated in more detail is how Omicron-based P2P systems can be protected from malicious peers, especially when these peers have been assigned crucial roles, i.e., Maintainers.

Further, the workload assigned to each role depends considerably on the various rates particular actions occur. E.g., frequent queries increases the routing workload while frequent peer arrivals and departures boost the maintenance overhead. Therefore, the workload for each role may vary significantly. The development of advanced incentive mechanisms for role assignment that consider the relevant attribute rates is an essential contribution.

Last but not least, the deployment of the presented architecture may reveal further hidden challenges that can motivate the development of more advanced solutions applicable to a variety of application scenarios.

Bibliography

- [AB02] R. Albert and A.-L. Barabasi. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74(47), 2002.
- [Abi04] B. N. Abilene. Internet2 netflow, weekly reports. <http://abilene.internet2.edu/>, 2004.
- [ACMD⁺03] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Puceva, R. Schmidt, and J. Wu. Advanced Peer-to-Peer Networking: The P-Grid System and its Applications. *PIK - Praxis der Informationsverarbeitung und Kommunikation, Special Issue on P2P Systems*, 26(3):86–89, 2003.
- [Ada99] L. A. Adamic. The Small World Web. In S. Abiteboul and A.-M. Veroustre, Editors, *Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, number 1696, Pages 443–452. Springer-Verlag, 1999.
- [Adl95] R. M. Adler. Distributed Coordination Models for Client/Server Computing. *Computer*, 28(4):14–22, 1995.
- [ADS02] J. Aspnes, Z. Diamadi, and G. Shah. Fault-tolerant Routing in Peer-to-Peer Systems. In *Proceedings of the 21th Annual Symposium on Principles of Distributed Computing*, Pages 223–232. ACM Press, 2002.
- [AFJ00] M. Arlitt, R. Friedrich, and T. Jin. Performance evaluation of Web proxy cache replacement policies. *Performance Evaluation*, 39(1-4):149–164, 2000.
- [AG01] O. Aalen and H. Gjessing. Understanding the Shape of the Hazard Rate: A Process Point of View. *Statistical Science*, 16(1):1–22, 2001.
- [Ald03] D. Aldous. A Stochastic Complex Network Model. *Electronic Research Announcements of the American Mathematical Society*, 9:152–161, 2003.
- [ALH02] L. A. Adamic, R. M. Lukose, and B. A. Huberman. *Handbook of Graphs and Networks: From the Genome to the Internet*, Chapter Local Search in Unstructured Networks. Wiley-VCH, Berlin, 2002.
- [ALPH01] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physical Review E*, 64(046135), 2001.

- [ALR00] A. Avizienis, J.-C. Laprie, and B. Randell. Fundamental Concepts of Dependability. In *Proceedings of the 3rd IEEE Information Survivability Workshop (ISW 2000)*, Pages 7–12, October 2000.
- [ANS00] ANSI/IEEE. 1516-2000 High Level Architecture (HLA), 2000.
- [AS03] J. Aspnes and G. Shah. Skip Graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, USA, 12–14 2003.
- [ATS04] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.
- [AW04] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Inc., 2nd Edition, 2004. ISBN: 0-471-45324-2.
- [AX02] A. Andrzejak and Z. Xu. Scalable, Efficient Range Queries for Grid Information Services. In *Proceedings of the Second IEEE International Conference on Peer-to-Peer Computing (P2P2002)*, September 2002.
- [Ban02] S. Banerjee. myns (P2P) simulator. <http://www.cs.umd.edu/suman/research/myns/>, 2002.
- [BB03] A. Barabasi and E. Bonabeau. Scale-Free Networks. *Scientific American*, 288(5):60–69, 2003.
- [BBR04] P. O. Boykin, J. S. Bridgewater, and V. Roychowdhury. Statistical Properties of Query Strings. Preprint, January 2004.
- [BCD⁺03] S. Bhattacharyya, E. Cheong, J. Davis, M. Goel, C. Hylands, B. Kienhuis, E. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Reekie, N. Smyth, J. Tsay, B. Vogel, W. Williams, Y. Xiong, Y. Zhao, and H. Zheng. Heterogeneous Concurrent Modeling and Design in Java: Ptolemy II Design. <http://ptolemy.eecs.berkeley.edu/papers/03/ptIIDesignDomains/>, 2003.
- [BDET00] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Pages 34–43. ACM Press, 2000.
- [Ber62] C. Berge. *The Theory of Graphs and its Applications*. John Wiley & Sons, Inc, 1st Edition, 1962. (Translated by Alison Doig).
- [BKK⁺03] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up Data in P2P Systems. *Communications of the ACM*, 46(2):43–48, 2003.
- [Blo70] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

-
- [BMM02] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems. In *Proceedings of the 22th International Conference on Distributed Computing Systems*, July 2002.
- [BMS93] H. Block, J. Mi, and T. Savits. Burn-In and Mixed Populations. *Journal of Applied Probability*, 30(3):692–702, 1993.
- [Bon02] E. Bonsma. Fully Decentralized, Scalable Look-up in a Network of Peers using Small World Networks. In *Proceedings of Systemics, Cybernetics and Informatics (SCI)*, 2002.
- [BQ03] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proceedings of the International Workshop on Web Content Caching and Distribution*, October 2003.
- [BRS02] A. R. Bharambe, S. Rao, and S. Seshan. Mercury: A Scalable Publish-Subscribe System for Internet Games. In *ACM Netgames*, April 2002.
- [BS97] H. Block and T. Savits. Burn-In. *Statistical Science*, 12(1):1–19, 1997.
- [BSGL96] F. Bernabei, V. D. Simone, L. Gratta, and M. Listanti. Shuffle vs. Kautz/De Bruijn Logical Topologies for Multihop Networks: a Throughput Comparison. In *Proceedings of the International Broadband Communications*, Pages 271–282, 1996.
- [BSS02] A. Brinkmann, K. Salzwedel, and C. Scheideler. Compact, Adaptive Placement Schemes for Non-Uniform Requirements. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, Pages 53–62. ACM Press, 2002.
- [BT80] W. Bridges and S. Toueg. On the impossibility of directed Moore graphs. *Journal of Combinatorial Theory Series B*, 29:339–341, 1980.
- [BWA96] J. W. Barrus, R. C. Waters, and D. B. Anderson. Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments. *IEEE Computer Graphics and Applications*, 16(6):50–57, November 1996.
- [Cai02] S. R. Cain. Distinguishing between lognormal and Weibull distributions [time-to-failure data]. *IEEE Transactions on Reliability*, 51(1):32–38, 2002.
- [CAN02] F. M. Cuenca-Acuna and T. D. Nguyen. Text-Based Content Search and Retrieval in ad hoc P2P Communities. In *Proceedings of International Workshop on Peer-to-Peer Computing*. Springer-Verlag, May 2002.
- [CBG03] F. Cecin, J. Barbosa, and C. Geyer. FreeMMG: An Hybrid Peer-to-Peer, Client-Server, and Distributed Massively Multiplayer Game Simulation Model. In *Proceedings of the 2nd Brazilian Workshop on Games and Digital Entertainment (WJogos'03)*, November 2003.
- [CDG⁺02] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Security for structured peer-to-peer overlay networks. In *Fifth Symposium on Operating Systems Design and Implementation (OSDI'02)*, December 2002.

- [CDKR02] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, 20(8):100–110, 2002.
- [CF04] C. Cramer and T. Fuhrmann. On the fundamental communication abstraction supplied by P2P overlay networks. *European Transactions on Telecommunications*, December 2004.
- [CLL02] J. Chu, K. Labonte, and B. N. Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. In *Proceedings of ITCom: Scalability and Traffic Control in IP Networks (SPIE)*, July 2002.
- [Con02] J. Considine. Cluster-based Optimizations for Distributed Hash Tables. Technical Report, 2003-031, CS Department, Boston University, November 2002.
- [CSWH00] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.
- [DAS⁺00] V. Darlagiannis, R. Ackermann, A. E. Saddik, N. Georganas, and R. Steinmetz. Suitability of Java for Virtual Collaboration. In *Proc. of Net.ObjectDays'2000*, Pages 71–78, October 2000.
- [Dat02] M. Datar. Butterflies and Peer-to-Peer Networks. In *Proceedings of ESA 2002 (LNCS)*, June 2002.
- [dB46] N. G. de Bruijn. A combinatorial problem. In *Proceedings of the Koninklijke Nederlandse Academie van Wetenschappen*, Pages 758–764, 1946.
- [DFM00] R. Dingledine, M. J. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In *Workshop on Design Issues in Anonymity and Unobservability*, Pages 67–95, 2000.
- [DG00] V. Darlagiannis and N. D. Georganas. Virtual Collaboration and Media Sharing using COSMOS. In *Proc. 4th WORLD Multiconference on Circuits, Systems, Communications & Computers (CSCC 2000)*, July 2000.
- [DGM02] N. Daswani and H. Garcia-Molina. Query-Flood DoS Attacks in Gnutella. In *Proceedings of the ACM Transactions on Information Systems (TOIS '02)*, 2002.
- [DGMY03] N. Daswani, H. Garcia-Molina, and B. Yang. Open Problems in Data-sharing Peer-to-Peer Systems. In *Proceedings of ICDT 2003*, January 2003.
- [DKK⁺01] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Pages 202–215. ACM Press, 2001.
- [DLH⁺05] V. Darlagiannis, N. Liebau, O. Heckmann, A. Mauthe, and R. Steinmetz. Caching Indices for Efficient Lookup in Structured Overlay Networks. In

- Proceedings of the Fourth International Workshop on Agents and Peer-to-Peer Computing*, July 2005.
- [DLM03] V. Darlagiannis, N. Liebau, and A. Mauthe. File sharing Scenario. Internal MMAPPS Deliverable, Technische Universität Darmstadt, July 2003.
- [DMLS04] V. Darlagiannis, A. Mauthe, N. Liebau, and R. Steinmetz. An Adaptable, Role-based Simulator for P2P Networks. In *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods*, Pages 52–59, June 2004.
- [DMS04] V. Darlagiannis, A. Mauthe, and R. Steinmetz. Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems. *Journal of Networks and System Management*, 12(3):371–395, 2004.
- [DMS05] V. Darlagiannis, A. Mauthe, and R. Steinmetz. Optimizing Overlay Network Stability using Burn-In Methods. Submitted for publication, March 2005.
- [Dow01a] A. B. Downey. Evidence for long-tailed distributions in the Internet. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, Pages 229–241, November 2001.
- [Dow01b] A. B. Downey. The Structural Cause of File Size Distributions. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, August 2001.
- [eDo05] eDonkey2000. <http://www.edonkey2000.com>, 2005.
- [EJ01] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). Internet RFC-3174, september 2001.
- [EM94] B. Elbert and B. Martyna. *Client/Server Computing: Architecture, Application, and Distributed System Management*. Artech House, 1st Edition, 1994.
- [Fel71] W. Feller. *An Introduction to Probability Theory and Its Applications: Volume II*. John Wiley & Sons, Ltd, 2nd Edition, 1971.
- [FG03a] P. Fraigniaud and P. Gauron. An Overview of the Content-Addressable Network D2B. In *Annual ACM Symposium on Principles of Distributed Computing*, July 2003.
- [FG03b] P. Fraigniaud and P. Gauron. The Content-Addressable Network D2B. Technical Report 1349, LRI, Univ. Paris-Sud, Paris, France, January 2003.
- [FL92] M. Fiol and A. Llado. The Partial Line Digraph Technique in the Design of Large Interconnection Networks. *IEEE Transactions on Computers*, 41(7):848–857, 1992.
- [FM02] M. J. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS-9)*, Washington, D.C., November 2002.

- [FM03] M. J. Freedman and D. Mazières. Sloppy Hashing and Self-Organizing Clusters. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA, February 2003.
- [FP01] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(4):392–403, 2001.
- [FS02] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, September 2002.
- [FV00] K. Fall and K. Varadhan. The ns Manual. <http://www.isi.edu/nsnam/ns/doc-stable/index.html>, 2000.
- [FV02] M. J. Freedman and R. Vingralek. Efficient Peer-to-Peer Lookup Based on a Distributed Trie. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, Cambridge, MA, March 2002.
- [FYdM84] M. Fiol, L. A. Yebra, and I. A. de Miquel. Line Digraph Iterations and the (d,k) Digraph Problem. *IEEE Transactions on Computers*, 33(5):400–403, 1984.
- [Gab99] X. Gabaix. Zipf’s law for Cities: An Explanation. *The Quarterly Journal of Economics*, 114(3):739–767, 1999.
- [Gam00] S. Gammill. Java Visualiser (Javis 2.0). <http://cs.baylor.edu/donahoo/NIUNet/javis.html>, 2000.
- [Gav01] C. Gavoille. Routing in Distributed Networks: Overview and Open Problems. *ACM SIGACT News - Special Interest Group on Automata and Computability Theory*, 32:36–52, 2001.
- [GBL⁺03] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, February 2003.
- [GDS⁺03] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of Peer-to-Peer File Sharing Workload. In *Proceedings of 19th ACM Symposium on Operating Systems Principles*, October 2003.
- [GEBF⁺03] L. Garces-Erce, E. Biersack, P. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical Peer-to-Peer Systems. In *Proceedings of Euro-Par*, August 2003.
- [GLMT01] W. Gong, Y. Liu, V. Misra, and D. Towsley. On the Tails of Web File Size Distributions. In *Proceedings of the 39th Allerton Conference on Communication, Control, and Computing*, October 2001.
- [Glo04] Software Engineering Institute, Technological Glossary, Carnegie Mellon University. <http://www.sei.cmu.edu/str/indexes/glossary/>, 2004.

- [GMS04] C. Gkantsidis, M. Mihail, and A. Saberi. Random Walks in Peer-to-Peer Networks. In *Proceedings of IEEE INFOCOM 2004*, March 2004.
- [Gnu05a] Gnutella. <http://www.gnutella.com>, 2005.
- [Gnu05b] Gnutella 2. <http://www.gnutella2.com>, 2005.
- [Gon02] L. Gong. Project JXTA: A Technology Overview, October 2002.
- [GPU01] P. R. Gopal Pandurangan and E. Upfal. Building Low-Diameter Peer-to-Peer Networks. In *Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2001.
- [GSGM03] P. Ganesan, Q. Sun, and H. Garcia-Molina. YAPPERS: A Peer-to-Peer Lookup Service Over Arbitrary Topology. In *Proceedings of IEEE Infocom*, 2003.
- [Has04] G. Hasslinger. Peer-to-Peer Networks: The View of Internet Service Providers, 2004.
- [HAY⁺05] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell. A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems. In *Proceedings of the IEEE International Conference on Information Technology (ITCC)*, April 2005.
- [Hec04] O. Heckmann. *A System-oriented Approach to Efficiency and Quality of Service for Internet Service Providers*. PhD Thesis, Technische Universität Darmstadt, Germany, December 2004.
- [HHH⁺02] M. Harren, J. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. Complex queries in DHT-based peer-to-peer networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.
- [HHL⁺03] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proceedings of VLDB'03*, September 2003.
- [HJS⁺03] N. J. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS '03)*, March 2003.
- [HK03] K. Hildrum and J. Kubiawicz. Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-Peer Networks. In *Proceedings of 17th International Symposium on Distributed Computing*, October 2003.
- [HKRZ02] K. Hildrum, J. D. Kubiawicz, S. Rao, and B. Y. Zhao. Distributed Object Location in a Dynamic Network. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, Pages 41–52. ACM Press, 2002.
- [HM98] F. Howell and R. McNab. Simjava: a discrete event simulation package

- p for Java with applications in computer systems modelling. In
- International Conference on Web-based Modelling and Simulation*
- , January 1998.
- [HMKR04] M. Hollick, I. Martinovic, T. Krop, and I. Rimac. A Survey on Dependable Routing in Sensor Networks, Ad hoc Networks, and Cellular Networks. In *Proceedings of the 30th EUROMICRO Conference (EUROMICRO 2004)*, Pages 495–502. IEEE Computer Society Press, Los Alamitos, September 2004.
 - [Hol04] M. Hollick. *Dependable Routing for Cellular and Ad hoc Networks*. PhD Thesis, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt, Germany, December 2004.
 - [How04] J. Howard. An Overview of the Andrew File System. In *Proceedings of the USENIX Winter Technical Conference*, Pages 23–26, January 2004.
 - [HSSS04] M. Hollick, J. Schmitt, C. Seipl, and R. Steinmetz. On the Effect of Node Misbehavior in Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications, ICC'04, Paris, France*, volume 6, Pages 3759–3763. IEEE, June 2004.
 - [HT00] A. Horvath and M. Telek. Approximating Heavy Tailed Behavior with Phase Type Distributions. In *Proceedings of 3rd International Conference on Matrix-Analytic Methods in Stochastic Models*, June 2000.
 - [HW97] F. Hsu and D. Wei. Efficient Routing and Sorting Schemes for de Bruijn Networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(11):1157–1170, 1997.
 - [HW02] S. Hazel and B. Wiley. Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, March 2002.
 - [HWBM02] C. Hoile, F. Wang, E. Bonsma, and P. Marrow. Core specification and experiments in DIET: a decentralised ecosystem-inspired mobile agent system. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, Pages 623–630. ACM Press, 2002.
 - [IHK04] T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multi-player Online Games. In *Proceedings of the NetGames, ACM SIGCOMM'04 Workshops*, August 2004.
 - [IRD02] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, Pages 213–222. ACM Press, 2002.
 - [ISO97] ISO/IEC. 14772: Information technology – Computer graphics and image processing – The Virtual Reality Modeling Language (VRML), 1997.

-
- [ISO99a] ISO/IEC. 14496-1: Information Technology – Coding of audio-visual objects, Part 1: Systems, 1999.
- [ISO99b] ISO/IEC. 14496-6: Information Technology – Coding of audio-visual objects, Part 6: Delivery Multimedia Integration Framework, 1999.
- [ITU94] T. S. S. ITU. Terms and Definitions related to Quality of Service and Network Performance including Dependability. Technical Report ITU-T Recommendation E.800, August 1994.
- [IUKB⁺04] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five months in a Torrent’s Lifetime. In *Proceedings of Passive and Active Measurements (PAM) 2004*, April 2004.
- [JCH84] R. Jain, D.-M. Chiu, and W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. Research Report TR-301 TR-301, DEC, September 1984.
- [JMT94] G. C. Jogesh Muppala and K. Trivedi. Stochastic Reward Nets for Reliability Prediction. *Communications in Reliability, Maintainability and Serviceability*, 1(2), July 1994.
- [Jos03] S. Joseph. An Extendible Open Source P2P Simulator. *P2P Journal*, November:1–15, 2003.
- [JP03] S. Jagannathan and G. Pandurangan. Stochastic Analysis of a Fault-Tolerant and Bandwidth-Efficient P2P Network. Technical Report TR-03-029, Purdue University, 2003.
- [Jun99] D. Jungnickel. *Graphs, Networks and Algorithms*. Springer, 1st Edition, 1999.
- [JW00] P. Jogalekar and M. Woodside. Evaluating the scalability of distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(6):589–603, 2000.
- [Kab01] M. Kabanov. In Defense of Gnutella. Online-Artikel, <http://www.gnutellameter.com/gnutella-editor.html>, 2001.
- [Kan02] J. Kangasharju. *Internet Content Distribution*. PhD Thesis, Department of Computer Science, University of Nice, Nice, France, April 2002.
- [Kau67] A. Kaufmann. *Graphs, Dynamic Programming and Finite Games*. Academic Press, 1st Edition, 1967. (Translated by Henry C. Sneyd).
- [KBC⁺00] J. Kubiawicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao. OceanStore: an Architecture for Global-scale Persistent Storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Pages 190–201. ACM Press, 2000.

- [KK83] W. Kuo and Y. Kuo. Facing the Headaches of Early Failures: A State-of-the-Art Review of Burn-In Decisions. *Proceedings of the IEEE*, 71(11):1257–1266, 1983.
- [KK03] F. Kaashoek and D. R. Karger. Koorde: A Simple Degree-optimal Hash Table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, February 2003.
- [Kle99] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. *Cornell Computer Science Technical Report*, 1776(99), 1999.
- [Kle02] J. Kleinberg. Small-World Phenomena and the Dynamics of Information. In T. G. Dietterich, S. Becker, and Z. Ghahramani, Editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
- [KLL⁺97] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, Pages 654–663. ACM Press, 1997.
- [KLXH04] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, March 2004.
- [Kul95] V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1st Edition, 1995.
- [KW04] H. Koubaa and Z. Wang. A Hybrid Content Location Approach between Structured and Unstructured Topology. In *Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop*, June 2004.
- [KWX01] B. Krishnamurthy, J. Wang, and Y. Xie. Early Measurements of a Cluster-based Architecture for P2P Systems. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop*, Pages 105–109. ACM Press, 2001.
- [LBK02] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Observations on the Dynamic Evolution of Peer-to-Peer Networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.
- [LCP⁺04] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Survey and Tutorial*, March 2004.
- [LDMS05] N. Liebau, V. Darlagiannis, A. Mauthe, and R. Steinmetz. Token-based Accounting for P2P-Systems. In *Proceedings of the Kommunikation in Verteilten Systemen KiVS 2005*, February 2005.
- [Lei91] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1991.

- [LHSH04] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS04)*, February 2004.
- [Liu90] Z. Liu. Optimal routing in the de Bruijn networks. In *Proceedings of the 10th International Conference on Distributed Computing Systems*, Pages 537–544, May 1990.
- [LKR04] J. Liang, R. Kumar, and K. Ross. The KaZaA Overlay: A Measurement Study. September 2004.
- [LKR03] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience. In *Proceedings of ACM SIGCOMM'03*, Pages 395–406, August 2003.
- [LN97] W. Litwin and M.-A. Neimat. LS*S: A High-Availability and High-Security Scalable Distributed Data Structure. In *Proceedings of Research Issues in Data Engineering (RIDE)*, 1997.
- [LNBK02] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the 21th Annual Symposium on Principles of Distributed Computing*, Pages 233–242. ACM Press, 2002.
- [LNS96] W. Litwin, M.-A. Neimat, and D. A. Schneider. LH*?a scalable, distributed data structure. *ACM Transactions on Database Systems (TODS)*, 21(4):480–525, 1996.
- [LRS02] Q. Lv, S. Ratnasamy, and S. Shenker. Can Heterogeneity Make Gnutella Scalable? In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, March 2002.
- [LRW03] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the KaZaa Network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, June 2003.
- [LS96] Z. Liu and T.-Y. Sung. Routing and Transmitting Problems in de Bruijn Networks. *IEEE Transactions on Computers*, 45(9):1056–1062, 1996.
- [LS03] C. Law and K.-Y. Siu. Distributed Construction of Random Expander Networks. In *Proceedings of IEEE INFOCOM 2003*, April 2003.
- [LSG⁺04] J. Li, J. Stribling, T. M. Gil, R. Morris, and M. F. Kaashoek. Comparing the performance of distributed hash tables under churn. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04)*, February 2004.
- [LXN04] Y. Liu, L. Xiao, and L. M. Ni. Building a Scalable Bipartite P2P Overlay Network. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, April 2004.
- [MAC04] K. S. Mikael Akerholm, Johan Fredriksson and I. Crnkovic. Quality Attribute Support in a Component Technology for Vehicular Software. In *Proceedings*

- of the Fourth Conference on Software Engineering Research and Practice in Sweden, October 2004.
- [Mar02] E. P. Markatos. Tracing a large-scale Peer-to-Peer System: an hour in the life of Gnutella. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and Grid*, Pages 65–74, May 2002.
 - [MBR03] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed Hashing in a Small World. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and System (USITS '03)*, March 2003.
 - [MH03] A. Mauthe and D. Hutchison. Peer-to-Peer Computing: Systems, Concepts and Characteristics. *Praxis in der Informationsverarbeitung und Kommunikation (PIK)*, 26(2):60–64, June 2003.
 - [Mi95] J. Mi. Bathtub Failure Rate and Upside-Down Bathtub Mean Residual Life. *IEEE Transactions on Reliability*, 44(3):388–391, 1995.
 - [Mi04] J. Mi. A General Approach to the Shape of Failure Rate and MRL Functions. *Naval Research Logistics*, 51(4):543–556, 2004.
 - [Mil67] S. Milgram. The small world problem. *Psychology Today*, 1:60–67, 1967.
 - [Min78] E. Minieka. *Optimization Algorithms for Networks and Graphs*. Macel Dekker Inc, 1st Edition, 1978.
 - [Mit03] M. Mitzenmacher. A Brief History of Generative Models for Power Law and Lognormal Distributions. *Internet Mathematics*, 1(2):226–251, 2003.
 - [Mit04] M. Mitzenmacher. Dynamic Models for File Sizes and Double Pareto Distributions. *Internet Mathematics*, 1(3):305–334, 2004.
 - [MM02] P. Maymounkov and D. Mazières. Kademlia: A Peer-to-peer Information System Based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.
 - [MNR02] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a Scalable and Dynamic Emulation of the Butterfly. In *Proceedings of the 21th Annual Symposium on Principles of Distributed Computing*, Pages 183–192. ACM Press, 2002.
 - [Mon04] A. Montresor. A Robust Protocol for Building Superpeer Overlay Topologies. In *In Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing*, August 2004.
 - [MS03a] J. Mischke and B. Stiller. Peer-to-Peer Overlay Network Management Through AGILE. In *Kluwer Academic Publishers, IFIP/IEEE International Symposium on Integrated Network Management (IM)*, March 2003.
 - [MS03b] J. Mischke and B. Stiller. Rich and Scalable Peer-to-Peer Search with SHARK. In *5th International Workshop on Active Middleware Services (AMS 2003)*, June 2003.

- [MT04] A. Mauthe and P. Thomas. *Professional Content Management Systems – Handling Digital Media Assets*. John Wiley & Sons, Ltd., 1st Edition, 2004.
- [MTG03] H. D. Meer, K. Tutschku, and P. T. Gia. Dynamic Operation of Peer-to-Peer Overlay Networks. *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 2003(2):65–73, 2003.
- [MY00] J.-W. Mao and C.-B. Yang. Shortest path routing and fault-tolerant routing on de Bruijn networks. *Networks*, 35(3):207–215, 2000.
- [Nap05] Napster. <http://www.napster.com>, 2005.
- [Neu81] M. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. John Hopkins University Press, Baltimore, 1st Edition, 1981. ISBN: 0-801-82560-1.
- [Neu94] B. C. Neuman. *Readings in Distributed Computing Systems*, Chapter Scale in Distributed Systems, Pages 463–489. IEEE Computer Society Press, 1994.
- [New00] M. E. J. Newman. Models of the Small World. *Journal of Statistical Physics*, 101(3/4):819–841, 2000.
- [NG01] W. H. Nicholas Gibbins. Scalability Issues for Query Routing Service Discovery. In *Proceedings of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS at the Fourth International Conference on Autonomous Agents (ICMAS2001)*, Pages 209–217, 2001.
- [NW03] M. Naor and U. Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach. In *Proceedings Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*, June 2003.
- [On04] G. On. *Quality of Availability for Widely Distributed and Replicated Content Stores*. PhD Thesis, Technische Universität Darmstadt, Germany, June 2004.
- [OOW93] E. O’Neil, P. O’Neil, and G. Weikum. The LRU-K Page Replacement Algorithm For Database Disk Buffering. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of data*, Pages 297–306, 1993.
- [Ora01] A. Oram. *Harnessing the Power of Disruptive Technologies*. O’Reilly, Sebastopol, CA, 2001.
- [Ove05] Overnet. <http://www.overnet.com>, 2005.
- [PGES04] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System. Technical Report PDS-2004-003, Delft University of Technology, April 2004.
- [PLC00] B. Page, T. Lechler, and S. Claassen. *Objektorientierte Simulation in Java mit dem Framework DESMO-J*. Libri Books on Demand, 2000.
- [Pot05] A. Potimeni. Simulative Study of the JXTA Virtual Network. Master Thesis, Darmstadt University of Technology, February 2005.

- [PRR97] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, Pages 311–320. ACM Press, 1997.
- [PSAS01] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne. The cost of peer discovery and searching in the Gnutella peer-to-peer file sharing protocol. In *Proceedings of the International Conference on Networks*, Pages 263–268, 2001.
- [Pug90] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.
- [RBDR⁺04] M. Roussopoulos, M. Baker, T. G. David Rosenthal, P. Maniatis, and J. Mogul. 2 P2P or Not P2P? In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04)*, February 2004.
- [RD90] J. Robinson and M. Devarakonda. Data cache management using frequency based replacement. In *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, Pages 134–142, 1990.
- [RD01a] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Pages 329–350, 2001.
- [RD01b] A. I. T. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In *Symposium on Operating Systems Principles*, Pages 188–201, 2001.
- [RDS04] A. Riska, V. Diev, and E. Smirni. Efficient Fitting of Long-tailed Data Sets into Phase-type Distributions. *Performance Evaluation Journal*, 55(1-2):147–164, 2004.
- [RFH⁺01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable Content Addressable Network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pages 161–172. ACM Press, 2001.
- [RFI02] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing Journal*, 6(1), Jan./Feb. 2002.
- [RH02] W. J. Reed and B. D. Hughes. From gene families and genera to incomes and internet file sizes: Why power laws are so common in nature. *Physical Review E*, 66(067103), December 2002.
- [Rim04] I. Rimac. *End-to-End Mechanisms for Rate-Adaptive Multicast Streaming over the Internet*. PhD Thesis, Technische Universität Darmstadt, Darmstadt, Germany, November 2004.

- [Rip01] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Proceedings of International Conference on Peer-to-peer Computing*, August 2001.
- [Rit01] J. Ritter. Why Gnutella Can't Scale - No, Really. Online-Article, <http://www.darkridge.com/jpr5/doc/gnutella.html>, 2001.
- [Riv92] R. Rivest. The MD5 Message-Digest Algorithm. Internet RFC-1321, April 1992.
- [RJ04] W. J. Reed and M. Jorgensen. The Double Pareto-Lognormal Distribution - A New Parametric Model for Size Distribution. *Communications in Statistics, Theory and Methods*, 33(8):1733–1753, January 2004.
- [RK02] S. Rhea and J. Kubiawicz. Probabilistic location and routing. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, June 2002.
- [RKP02] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding Good Peers in the Peer-to-Peer Networks. In *Proceedings of International Parallel and Distributed Computing Symposium (IPDPS)*, April 2002.
- [RKY⁺02] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a Geographic Hash Table for Data-centric Storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Pages 78–87. ACM Press, 2002.
- [RLA00] D. Roselli, J. R. Lorch, and T. E. Anderson. A Comparison of File System Workloads. In *Proceedings of the 2000 USENIX Annual Technical Conference*, June 2000.
- [RRK03] S. Rhea, T. Roscoe, and J. Kubiawicz. Structured Peer-to-Peer Overlays Need Application-Driven Benchmarks. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, February 2003.
- [RRVV01] R. Rajaraman, A. Richa, B. Voeking, and G. Vuppuluri. A Data Tracking Scheme for General Networks. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, July 2001.
- [RSS02] S. Ratnasamy, S. Shenker, and I. Stoica. Routing Algorithms for DHTs: Some Open Questions. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.
- [SAZ⁺04] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. *IEEE/ACM TRANSACTIONS ON NETWORKING*, 12(2):205–218, 2004.
- [Sch01] R. Schollmeier. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, 2001.

- [Sch05] R. Schollmeier. *Signaling and Networking in Unstructured Peer-to-Peer Networks*. PhD Thesis, Munich University of Technology, Germany, March 2005.
- [SCKH04] O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag. Organization, development and function of complex brain networks. *Trends in Cognitive Sciences*, 8(9):418–425, 2004.
- [SGG02] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.
- [Sin92] A. Sinha. Client-server computing. *Communications of the ACM*, 35(7):77–98, 1992.
- [SK03] R. Schollmeier and G. Kunzmann. GnuViz - Mapping the Gnutella Network to its Geographical Locations. *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 2003(2):74–79, 2003.
- [SL00] C. Shields and B. N. Levine. A protocol for anonymous communication over the Internet. In *Proceedings of the 7th ACM conference on Computer and communications security*, Pages 33–42. ACM Press, 2000.
- [SL04] A. Singh and L. Liu. A Hybrid Topology Architecture for P2P Systems. In *Proceedings of the 13th International Conference on Computer Communications and Networks*, October 2004.
- [SM02] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, Cambridge, MA, March 2002.
- [SMK⁺01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pages 149–160. ACM Press, 2001.
- [SMLN⁺03] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications. *IEEE Transactions on Networking*, 11(1):17–32, February 2003.
- [SMZ02] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location and Retrieval in Peer-to-Peer Systems by Exploiting Locality in Interests. *ACM SIGCOMM Computer Communication Review*, 32(1):80–80, 2002.
- [SN93] X.-H. Sun and L. M. Ni. Scalable problems and memory-bounded speedup. *Journal of Parallel and Distributed Computing*, 19(1):27–37, 1993.
- [SN04a] R. Steinmetz and K. Nahrstedt. *Multimedia Applications*. Springer, Berlin. Springer, 1st Edition, 2004.

-
- [SN04b] R. Steinmetz and K. Nahrstedt. *Multimedia Systems*. Springer, Berlin. Springer, 1st Edition, 2004.
- [SP03] J. Shneidman and D. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, February 2003.
- [SR94] K. Sivaraman and R. Ramaswami. Lightwave networks based on de Bruijn graphs. *IEEE/ACM Transactions on Networking (TON)*, 2(1):70–79, 1994.
- [SR04] D. Stutzbach and R. Rejaie. Towards a Better Understanding of Churn in Peer-to-Peer Networks. Technical Report CIS-TR-04-06, Department of Computer Science, University of Oregon, November 2004.
- [SSDN02] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP - Hypercubes, Ontologies and P2P Networks. In *Proceedings of the Agents and Peer-to-Peer Systems*, July 2002.
- [Str04] B. Strulo. Middleware to Motivate Co-operation in Peer-to-Peer Systems. *P2P Journal*, 2(1):589–603, 2004.
- [SW04a] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *IEEE/ACM Transactions on Networking*, 12(2):219–232, April 2004.
- [SW04b] R. Steinmetz and K. Wehrle. Peer-to-Peer-Networking and -Computing. *Informatik Spektrum, Aktuelles Schlagwort*, 27(1):51–54, 2004.
- [SW05] R. Steinmetz and K. Wehrle, Editors. *Peer-to-Peer Systems- and Applications*. Number LNCS 3485. Springer, 1st Edition, 2005.
- [TAA⁺03] B. Traversat, A. Arora, M. Abdelaziz, M. Duigou, C. Haywood, J.-C. Hugly, E. Pouyoul, and B. Yeager. Project JXTA 2.0 Super-Peer Virtual Network. <http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf>, May 2003.
- [TD03] N. S. Ting and R. Deters. 3LS-A p2p network simulator. In *Poster in The Third IEEE International Conference on Peer-to-Peer Computing*, September 2003.
- [TN97] P. Triantafillou and C. Neilson. Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems. *IEEE Transaction in Software Engineering*, 23(1):35–55, January 1997.
- [Tut04] K. Tutschku. A Measurement-Based Traffic Profile of the eDonkey Filesharing Service. In *Proceedings of the 5th Annual Passive & Active Measurement Workshop*, April 2004.
- [Tvr94] P. Tvrđik. Necklaces and scalability of Kautz digraphs. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, October 1994.
- [Tvr99] P. Tvrđik. Interconnection networks. Lecture notes, Spring 1999.

- [TXM02] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information Retrieval in Structured Overlays. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)*, October 2002.
- [Tya02] H. Y. Tyan. Design, Realization, and Evaluation of a Component-based Compositional Software Architecture for Network Simulation. http://www.j-sim.org/whitepapers/tyan_thesis.pdf, 2002.
- [WC03] X. F. Wang and G. Chen. Complex networks: Small-World, Scale-Free and Beyond. *IEEE Circuits and Systems Magazine*, 3(1):6–20, 2003.
- [WM01] M. Waldman and D. Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, Pages 126–135. ACM Press, 2001.
- [WRC00] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System. In *Proc. 9th USENIX Security Symposium*, Pages 59–72, August 2000.
- [XCRK03] D. Xu, H.-K. Chai, C. Rosenberg, and S. Kulkarni. Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN 2003)*, January 2003.
- [YGM01] B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *Proceedings of Very Large Databases (VLDB)*, 2001.
- [YGM02] B. Yang and H. Garcia-Molina. Improving Search in Peer-to-Peer Networks. In *22nd International Conference on Distributed Computing Systems (ICDCS'02)*, July 2002.
- [ZGG02] H. Zhang, A. Goel, and R. Govindan. Using the Small-World Model to Improve Freenet's Performance. In *Proceedings of IEEE Infocom*, 2002.
- [ZH03] R. Zhang and Y. C. Hu. Borg: a hybrid protocol for scalable application-level multicast in peer-to-peer networks. In *Proceedings of the 13th International Workshop on Network and Operating System Support for Digital Audio and Video*, Pages 172–179, June 2003.
- [ZHS⁺04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

Author's Publications

Journal Articles

1. Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems. *Journal of Network and Systems Management, Special Issue on Distributed Management*, 12(3):371-395, September 2004.
2. Bob Briscoe, Vasilios Darlagiannis, Oliver Heckmann, Oliver Huw, Vasilios Siris, Burkhard Stiller, and David Songhurst. A Market Managed Multi-Service Internet. *Computer Communications*, 26(4):405-415, March 2003.
3. Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. Optimizing Overlay Network Stability using Burn-In Methods, Submitted for publication to *Performance Evaluation*, Elsevier, March 2005.

Patents (Applications)

4. Nicolas Liebau, Vasilios Darlagiannis, and Andreas Mauthe. Dezentrales, token-basiertes Accountingsystem für autonome, verteilte Systeme. Patent Application No.: 04 101 386.3, August 2003.

Conference Contributions

5. Nicolas Liebau, Vasilios Darlagiannis, Oliver Heckmann, and Ralf Steinmetz. Asymmetric Incentives in Peer-to-Peer Systems. In *Proceeding of Americas Conference on Information Systems (AMCIS 2005)*, "Economics of Peer-to-Peer Networks" Mini-track, August 2005.
6. Vasilios Darlagiannis, Nicolas Liebau, Oliver Heckmann, Andreas Mauthe and Ralf Steinmetz. Caching Indices for Efficient Lookup in Structured Overlay Networks. In *Proceeding of the Fourth International Workshop on Agents and Peer-to-Peer Computing*, July 2005.

7. Nicolas Liebau, Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. Token-based Accounting for P2P-Systems. In *Proceeding of Kommunikation in Verteilten Systemen KiVS 2005*, pages 16-28, February 2005. (**Best Paper Award**).
8. Vasilios Darlagiannis, Andreas Mauthe, Nicolas Liebau, and Ralf Steinmetz. Distributed Maintenance of Mutable Information for Virtual Environments. In *Proceedings of 3rd IEEE International Workshop on Haptic Audio Visual Environments and their Applications - HAVE'04*, pages 87-92, October 2004.
9. Vasilios Darlagiannis, Andreas Mauthe, Nicolas Liebau, and Ralf Steinmetz. An Adaptable, Role-based Simulator for P2P Networks. In *Proceedings of International Conference on Modeling, Simulation and Visualization Methods*, Las Vegas, Nevada, USA, pages 52-59, June 2004.
10. Vasilios Darlagiannis, Martin Karsten, and Ralf Steinmetz. Burst Shaping Queueing. In *Proceedings of Computer Networks and Distributed Systems (WMC)*, pages 65-70. SCS, January 2003. ISBN 1-56555-261-X.
11. Oliver Heckmann, Vasilios Darlagiannis, Martin Karsten, and Ralf Steinmetz. A Price Communication Protocol for a Multi-Service Internet. In *Proceedings of Informatik 2001 - Wirtschaft und Wissenschaft in der Network Economy - Visionen und Wirklichkeit (GI/OCG 2001)*, September 2001.
12. Mojtaba Hosseini, Vasilios Darlagiannis, and Nicolas Georganas. COSMOS2: Interactive Collaborative Virtual Environment Using MPEG-4. In *Proceedings of the 8th International Conference on Advances in Communications and Control*, Crete, Greece, June 2001.
13. Ralf Ackermann, Vasilios Darlagiannis, Manuel Goertz, Martin Karsten, and Ralf Steinmetz. An Open Source H.323-SIP Gateway as Basis for Supplementary Service Interworking. In *Proceedings of the 2nd IP Telephony Workshop*, New York, pages 169-175, April 2001.
14. Vasilios Darlagiannis, Ralf Ackermann, Abdulmotaleb El Saddik, Nicolas Georganas, and Ralf Steinmetz. Suitability of Java for Virtual Collaboration. In *Proceedings of Net.ObjectDays 2000*, Erfurt Germany, pages 71-78, October 2000.
15. Ralf Ackermann, Vasilios Darlagiannis, Utz Roedig, and Ralf Steinmetz. Using DMIF for abstracting from IP-Telephony Signaling Protocols. In *Proceedings of the Seventh International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS 2000)*, Enschede, The Netherlands, pages 104-115, October 2000.
16. Vasilios Darlagiannis and Nicolas Georganas. Virtual Collaboration and Media Sharing using COSMOS. In *Proceedings of the 4th World Multiconference on Circuits, Systems, Communications & Computers (CSCC 2000)*, Athens, Greece, July 2000.
17. Yiannis Mavraganis, Yiannis Maragoudakis, Nikos Pappas, Georgia Kyriakaki, Vasilios Darlagiannis, Reinhard Lueling, Fernando Cortes, M.Herreo, and K.Meyer. The SICMA Teleteaching Trial on ADSL and Intranet Networks. In *Proceedings of the*

European Conference on Multimedia Applications, Services and Techniques (EC-MAST 99), Madrid, Spain, May 1999.

Internet Drafts

18. Oliver Heckmann, Vasilios Darlagiannis, Martin Karsten, Ralf Steinmetz, and Bob Briscoe. Tariff Distribution Protocol (TDP). Internet Draft, March 2002. draft-heckmann-tdp-00.txt.

Book Chapters

19. Vasilios Darlagiannis. Hybrid Peer-to-Peer Systems. Contribution to Peer-to-Peer Systems- and Applications (editors Ralf Steinmetz and Klaus Wehrle), LNCS 3485, Springer, 2005.
20. Oliver Heckmann, Nicolas Liebau, Vasilios Darlagiannis, Axel Bock, Andreas Mauthe, and Ralf Steinmetz. From Integrated Publication and Information Systems to Information and Knowledge Environments: Essays Dedicated to Erich J. Neuhold on the Occasion of His 65th Birthday, volume 3379 of Lecture Notes in Computer Science, chapter A Peer-to-Peer Content Distribution Network, pages 69-78. Springer-Verlag GmbH, January 2005.

Theses

21. Vasilios Darlagiannis. COSMOS: Collaborative System based on MPEG-4 Objects and Streams. University of Ottawa, Ottawa, Ontario, Canada, December 1999.
22. Vasilios Darlagiannis. Design and Implementation of the Session Gateway, the Stream and the Download Services at the KYDONIA Multimedia Information Server, compliant to the DAVIC Standard. Diploma Thesis, Technical University of Crete, Chania, Greece, 1997.

Technical Reports, Project deliverables and Miscellaneous

23. Nicolas Liebau, Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. A Token-based Accounting Scheme for P2P-Systems. Technical Report TR-2004-05, Technische Universität Darmstadt, January 2004.
24. Vasilios Darlagiannis and David Hausheer. Design, Scalability, and Application of Peer-to-Peer Overlay Networks. Deliverable 5.1 of the MMAPPS EU Project IST-2001-34201, August 2004.

25. Vasilios Darlagiannis, Nicolas Liebau and Andreas Mauthe. Specification and implementation of Peer-to-Peer Adaptation Layer (Final). Deliverable 12 of the MMAPPS EU Project IST-2001-34201, January 2004.
26. Nicolas Liebau, Vasilios Darlagiannis, Andreas Mauthe. A Token-based Accounting Scheme for P2P-Systems. Technical Report TR-2004-05, Technische Universität Darmstadt, January 2004.
27. Vasilios Darlagiannis, Nicolas Liebau and Andreas Mauthe. Specification and implementation of Peer-to-Peer Adaptation Layer (Initial). Deliverable 11 of the MMAPPS EU Project IST-2001-34201, April 2003.
28. Martin Karsten, Oliver Heckmann and Vasilios Darlagiannis. Integrated Pricing System and Network Technology. Deliverable 5.1 of the M3I EU IST-1999-11429, June 2001
29. Ralf Ackermann, Vasilios Darlagiannis, and Ralf Steinmetz. Project Tenovis IP-TEL: SIP Integration in an H.323 based PBX. Technical Report Project Phase 3: Prototype System and SIP End-System for Test, Multimedia Communications (KOM), Darmstadt University of Technology, May 2000.
30. Ralf Ackermann, Vasilios Darlagiannis, Utz Rödig, and Ralf Steinmetz. Project Tenovis IPTEL: SIP Integration in an H.323 based PBX. Technical Report Project Phase 4: Final Report, Multimedia Communications (KOM), Darmstadt University of Technology, November 2000.
31. Vasilios Darlagiannis , Yannis Mavraganis, Yannis Maragioudakis, Georgia Kyriakaki, Nicos Pappas, and Chrysa Tsinaraki, The Data and Access Management Layer of a DAVIC-compliant Server (SICMA Server), Report on the activity D6 of the SICMA Project (ACTS 071), December 1996.
32. Stavros Christodoulakis, Vasilios Darlagiannis, Dora Magoulioti, Kyriakos Poulos, Fenia Zioga, Final Report on the MILORD Project, Report on the activity D9 of the MILORD project (AIM A2024), 1995.

Curriculum Vitae

Personal Details

Name: Vasilios Darlagiannis
Date of birth: 08.06.1973
Nationality: Greek

Education

1998 – 1999	University of Ottawa, Ottawa, Ontario, Canada School of Information Technology & Engineering Degree: Master in Electrical & Computer Engineering (M.A.Sc.)
1991 – 1997	Technical University of Crete, Chania, Greece Department of Electronic & Computer Engineering Degree: Diploma of Engineering (Dipl.-Ing.)

Awards and Scholarships

2005	Best paper award in the proceedings of KiVS'05
1998	International Scholarship from the University of Ottawa
1998	Admission Scholarship from the University of Ottawa
1992	Award for excellent academic performance from IKY (Greek Scholarships Organization)
1992	Award for excellent academic performance from ELKE (Greek Award Organization)
1989	Award for excellent academic performance from the Greek Ministry of Education

Professional Experience

01/00 – present	Full-time Research Assistant Multimedia Communications Lab (KOM) Darmstadt University of Technology
05/98 – 12/99	Full-time Research Assistant MCRLab University of Ottawa
02/96 – 02/98	Full-time Research Assistant Multimedia Systems Institute of Crete (MUSIC) Technical University of Crete
02/96 – 10/97	Part-time Research Assistant Multimedia Systems Institute of Crete (MUSIC) Technical University of Crete

Teaching Experience

Spring 2000 – Winter 2004	Teaching Assistant for KN-II Practice on Networking Technologies (every semester) Covered topics: <ul style="list-style-type: none">◇ Unix networking, sockets, RPCs,◇ Multi-threaded programming in C and Java,◇ P2P systems development based on JXTA,◇ Context-based communications Darmstadt University of Technology
Fall 1999	Teaching Assistant for ELG 2181 "Digital Computer Organization" University of Ottawa
Winter 1999	Teaching Assistant for ELG 4102 "Microwave & Optical Circuits & Component Design" University of Ottawa
Winter 1999	Teaching Assistant for ELG 4104 "Antennas and Propagation" University of Ottawa
Fall 1998	Teaching Assistant for ELG 2181 "Digital Computer Organization" University of Ottawa

Supervised Student, Diploma, and Master Theses

1. Alexander Hanaoka. Effective P2P Infrastructures for Multi Description/Multi Layered Video. *Student Thesis KOM-D-224*, May 2005.
2. Amulya Potimeni. Simulative Study of the JXTA Virtual Network. *Master Thesis KOM-D-224*, February 2005.
3. Tillmann Steinbrecher. JXTA-based Peer to Peer Credit System. *Student Thesis KOM-S-152*, July 2003.
4. Philipp Hielscher. Classification of Congestion Control Mechanisms. *Student Thesis KOM-S-134*, March 2002.
5. Idris Akar. Modeling and Simulation of mobile packet switched Networks. *Student Thesis KOM-D-164*, February 2002.
6. Jarno Gassenbauer. Performance Analysis & Optimization of Multiple TCP Connections over a Packet Switched Mobile Network. *Student Thesis KOM-D-162*, January 2002.

Appendix A

Discrete and Continuous Distributions

A.1. Weibull distribution

A random variable is said to have a Weibull distribution if its distribution is given, for some $\lambda > 0, \eta > 0$ by:

$$f(x) = \lambda \eta x^{\eta-1} e^{-\lambda x^\eta}. \quad (\text{A.1})$$

The CDF of the Weibull distribution is provided by:

$$F(x) = 1 - e^{-(\lambda x)^\eta}. \quad (\text{A.2})$$

A.2. Power law and Pareto distribution

A non-negative random variable X is said to have *power law* distribution if:

$$Pr\{X \geq x\} = cx^{-\alpha}, \quad (\text{A.3})$$

for constants $c > 0$ and $\alpha > 0$. In a power law distribution asymptotically the tails fall according to the power α . Such a distribution leads to much heavier tails than other common models, such as exponential distributions. One specific commonly used power law distribution is the *Pareto* distribution, which satisfies:

$$Pr\{X \geq x\} = \left(\frac{x}{k}\right)^{-\alpha}, \quad (\text{A.4})$$

for some $\alpha > 0$ and $k > 0$. The Pareto distribution requires $X \geq k$. The density function for the Pareto distribution is $f(x) = \alpha k^\alpha x^{-\alpha-1}$. For a power law distribution, usually α falls in the range $0 < \alpha \leq 2$, in which case, X has infinite variance. If $\alpha \leq 1$, then X also has infinite mean [Mit03].

A.3. Lognormal distribution

A random variable X has a lognormal distribution if the random variable $Y = \ln X$ has a normal (i.e., Gaussian) distribution. Recall that the normal distribution Y is given by the density function:

$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\mu)^2}{2\sigma^2}}, \quad (\text{A.5})$$

where μ is the mean, σ is the standard deviation (σ^2 is the variance), and the range is $-\infty < y < \infty$. The density function for a lognormal distribution therefore satisfies:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(\log(x)-\mu)^2}{2\sigma^2}}. \quad (\text{A.6})$$

Note that the change of variables introduces an additional $1/x$ term outside of the exponential term. The corresponding complementary cumulative distribution function for a lognormal distribution is given by:

$$Pr\{X \geq x\} = \int_{z=x}^{\infty} \frac{1}{\sqrt{2\pi}\sigma z} e^{-\frac{(\log(z)-\mu)^2}{2\sigma^2}} dz. \quad (\text{A.7})$$

It can be stated that X has parameters μ and σ^2 when the associated normal distribution Y has mean μ and variance σ^2 , where the meaning is clear. The lognormal distribution is skewed, with mean $e^{\mu+\sigma^2/2}$, median e^μ , and mode $e^{\mu-\sigma^2}$. A lognormal distribution has finite mean and variance, in contrast to the power law distribution under natural parameters [Mit03].

Appendix B

Porting Chord Algorithms into the Open Architecture P2P Simulator

Chord⁴⁷ [SMLN⁺03] is a well known DHT-based structured overlay network. In many cases it is the reference point to evaluate the performance of novel approaches. Apparently, Omicron performance has been also evaluated against Chord.

However, the original version of Chord [SMK⁺01] had several unaddressed issues that were considered as "holes" in its design. Some of them have been addressed in the latest journal publication of Chord [SMLN⁺03]. In this section, the algorithmic description of the ported Chord in the open architecture simulator for P2P overlay networks is provided. The functionality of Chord has been divided to fit the the core identified roles. The original functionality required merely the instantiation of the Router, the Indexer and the Maintainer roles. However, in the extended version of Chord [DLH⁺05] the semantics of the Routing procedure have been modified and the Cacher role instantiated.

The pseudocode describing the processing of incoming lookup messages on each peer is provided in Algorithm B.0.1. The objects indexer, router and cacher are instantiations of the corresponding roles for the local peer. Moreover, lookup messages have been extended to include the *"KeyMeaning"* (e.g. looking for a successor, a finger, a resource, etc.) in addition to the required key value. The extended caching related operations are described

⁴⁷Refer to Section 3.2.1 for more details on Chord.

in Chapter 5.

Algorithm B.0.1: *CHORDROUTER::PROCESSLOOKUPMSG(msg)*

```

initiator = msg.getInitiator()
if (indexer.isResponsible(msg.getKey()))
    then {processLookupReply(msg)
entry = cache.getEntry(msg.getKey())
if (entry.getValue() not null)
    then {
        entry.updateUsageTimestamp()
        msg.setValue(getSuccessor())
        msg.setSender(getGuid())
        msg.setDestination(initiator)
        sendMessage(msg)
    }
return
if (initiator = getGuid())
    then {processLookupReply(msg)

else if (msg.getKeyMeaning = PREDECESSOR)
    then {
        msg.setValue(getPredecessor())
        msg.setSender(getGuid())
        msg.setDestination(initiator)
        sendMessage(msg)

        if (msg.getKey().between(getGuid(), getSuccessor(0)))
            then {
                if (msg.getMeaning() = NEW_PEER)
                    then {maintainer.accept(msg.getKey(), getSuccessor())

                else if (msg.getMeaning() = FINGER or
                    msg.getMeaning() = ITEM)
                    then {
                        msg.setValue(getSuccessor())
                        msg.setSender(getGuid())
                        msg.setDestination(initiator)
                        sendMessage(msg)

            else {
                dstLink = findClosestNeighbor(msg.getKey())
                msg.setSender(getPeer().getGuid())
                msg.setDestination(dstLink.getDestinationID())
                sendMessage(msg)
            }
        }
    }

```

Similarly, the pseudocode of the procedure for processing the reply to a lookup message

is provided in Algorithm B.0.2.

Algorithm B.0.2: *CHORDROUTER::PROCESSLOOKUPREPLYMSG(msg)*

```

if (msg.getKeyMeaning = PREDECESSOR)
  then { addSuccessor(msg.getValue())
        maintainer.notify(msg.getValue(), NOTIFY_PREDECESSOR)

  else if (msg.getKeyMeaning = ITEM)
    then { entry = cacher.isQueryPending(msg.getKey())
          if (entry! = null)
            then { caler.fillCacheEntry(entry)

    else if (msg.getKeyMeaning = FINGER)
      then { maintainer.fixFinger(msg.getKey(), msg.getValue())

```


Appendix C

Basic Concepts of Collaborative Virtual Environments

Collaborative Virtual Environments (CVEs) have been developed into an interesting research topic during the last decade. International bodies have specified standards to model their content such as VRML [ISO97], MPEG-4 Systems [ISO99a] or their architecture such as High Level Architecture [ANS00].

The media sources that compose the rendered scenes in CVEs have heterogeneous characteristics and delivery requirements. Two general media types may describe the content: *naturally captured media sources* and *synthetic graphics media sources*. Synthetic media are described by a number of hierarchically correlated objects (e.g. using the Virtual Reality Modeling Language (VRML) or the Binary Format for Scenes (BIFS) [ISO99a]). While VRML and BIFS describe the objects that compose the rendered scenes, the rendering itself is implemented with graphics packages such as OpenGL or Java3D [DAS⁺00].

CVEs may result in complex systems that are very costly to maintain and operate in presence of large users population. A useful mechanism to cope with the scalability problem is the concept of *Locales* [BWA96]. Locales are defined as integral partitions of the whole world. This way they provide more manageable portions of content to deal with. Neighbor Locales are linked to enable the transition among them.

Similarly, users' ability to experience the surrounding world is limited to reflect reality. In most approaches, a so-called *Area of Interest* (AoI) is defined as the spatial area of the world that a user can experience. In many cases, AoI is equivalent to the Locale that intersects with their position (additionally could be also the neighbor Locales). Users' presence in the virtual worlds is modeled using *avatars*, a graphic representation.

In many cases Virtual Environments contain intelligent entities that interact with the users actively when they approach them. Such entities are called *Bots* (or *Non Player Characters* (NPCs) in game terminology). NPCs differ from normal objects (e.g. those that compose the landscape of the virtual world) in two main aspects: (i) they have a state that can be modified in an interactive way by the users and (ii) they can stimulate events that are experienced by users in the vicinity. In addition to NPCs, a number of passive objects might have a state that could be modified by user actions. Such objects are called *mutable*.

Figure C.1 illustrates the aforementioned concepts. In this figure only neighbor Locales are connected, however, based on the rules that govern the virtual world, non-neighbor Locales

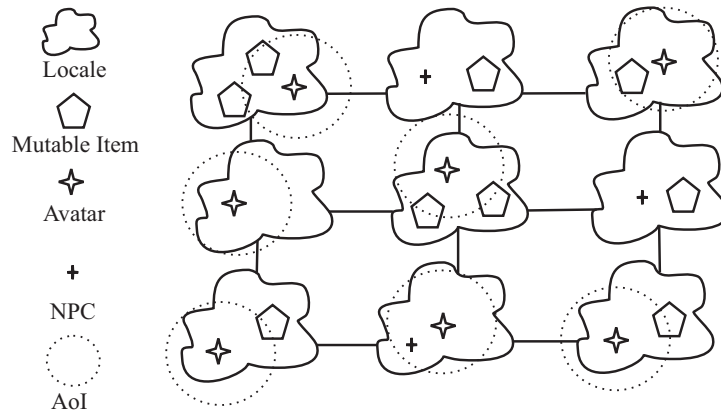


Figure C.1.: Modeling Concepts in Virtual Environments.

can be connected as well (i.e. through teleport devices). Moreover, mutable objects can belong to more than one Locales or can be moved to any of them.

CVEs are usually composed of multiple media sources such as synthetic graphics, animations, audio and video. Standardized documents such as MPEG-4 Systems [ISO99a] describes the scene composition specifications. Moreover, MPEG-4 DMIF [ISO99b] describes an integration framework to deliver the media sources to the destination end-points. COSMOS [DG00] is a reference implementation of the aforementioned specifications that integrates the rendering of synthetic graphics, animations, audio and video and employs multicast mechanisms to deliver the content among the collaborating end-points.

Appendix D

Additional Simulative Results

This appendix provides additional results to supplement those discussed in Section 8.4. The network size of the larger experiments is 65,536 peers. The results are shown in Figure D.1. Here, one can observe a similar query routing workload distribution as the one provided in Figure 8.5. The greedy closest neighbor policy shows a much larger standard deviation compared to the random neighbor policy and the restricted closest neighbor policy. It should be remarked that the y -axis scales logarithmically in order to provide a more comprehensive view of the results.

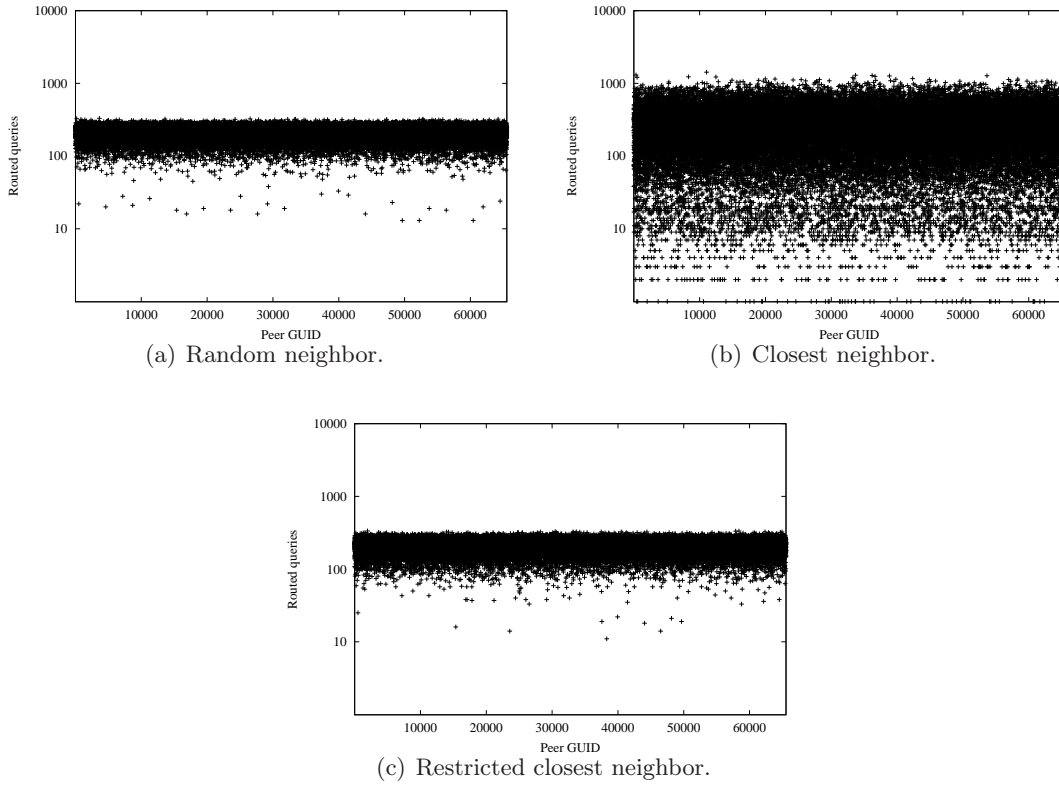


Figure D.1.: Query routing load distribution on large networks.

The results shown in Figure D.2 supplement the results described in Figure 8.10. The network configuration parameters are different, though. In this experiment, 4,096 clusters have been constructed. Also, the inter-cluster degree has been set to 4.

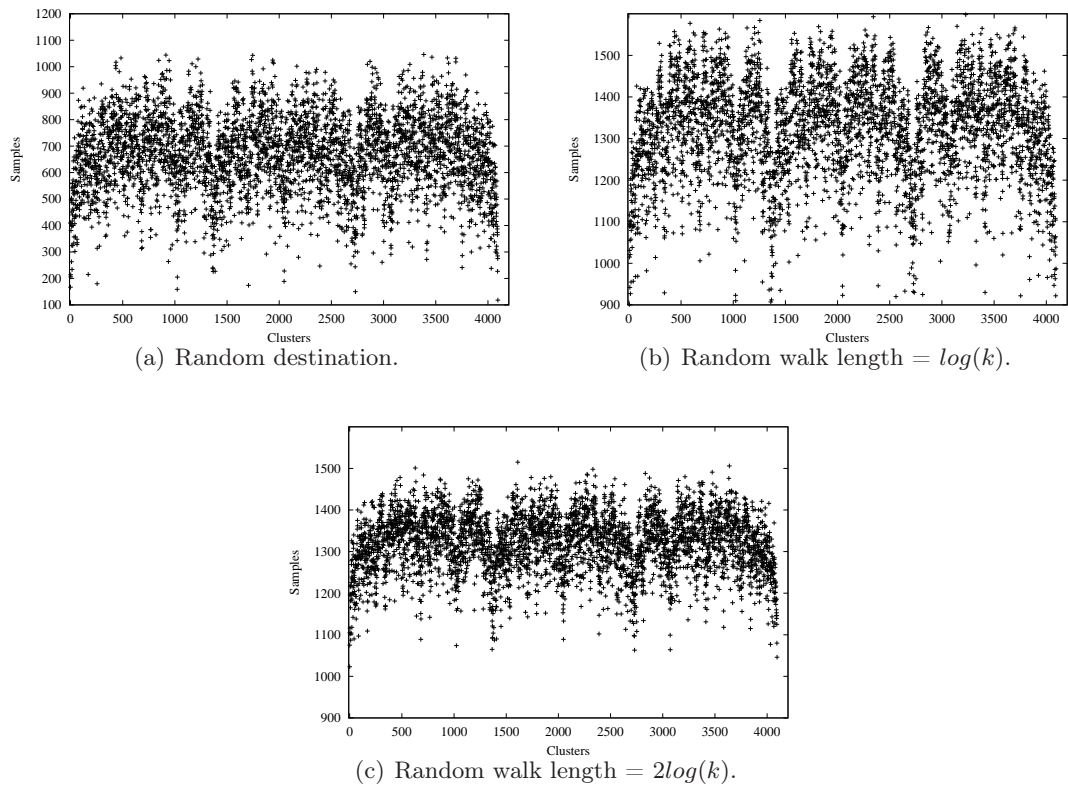


Figure D.2.: Sampling distribution using random walks.

Appendix E

Simulator Performance Evaluation

Although the performance of a simulator is highly related to the details of the embedded model, the efficiency of the implementation itself plays an important role for the acceptance of the tool. Using profiling tools, the implementation and in some cases the design of the tool has been improved. Figure E.1 provides two graphs that demonstrate the resource consumption during a number of experiments. As it can be easily seen from those graphs, both the memory consumption and the execution time increase linearly with respect to the size of the topology. This is very important and proves to a certain degree the validity of the implementation. Initially, J-Sim was selected to model the details of the underlying network. Although this solution offered a rich model, the performance was heavily decreased. The analysis of the results of profiling the used resources while using the simulator showed that approximately 75% of the CPU and memory was consumed by the J-Sim component. Thus, this led to the development of the lightweight approach described in Section 7.2.3. The selection of the heavy J-Sim based solution is still available in cases where a very detailed model is necessary. However, in order to evaluate better the

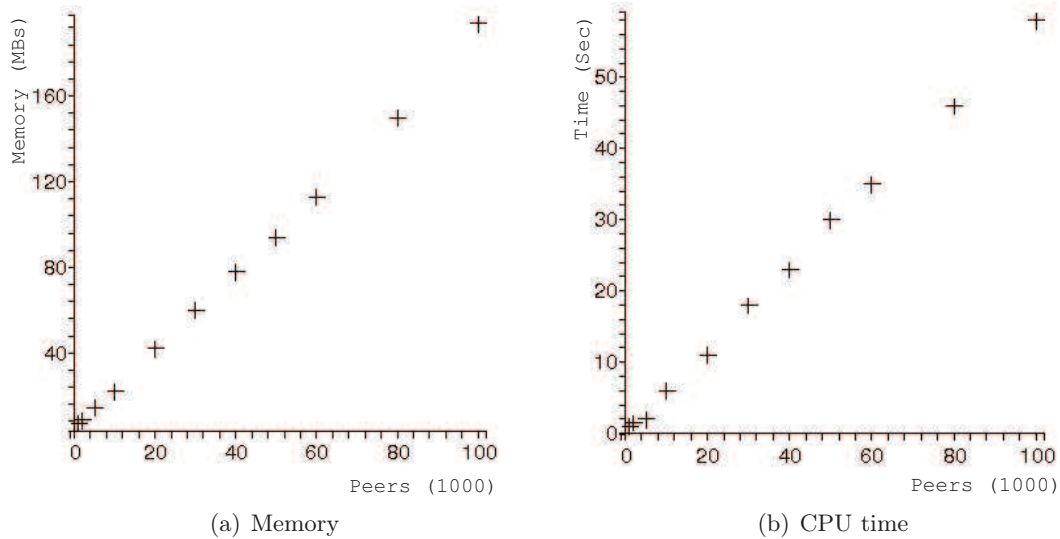


Figure E.1.: Simulator Performance.

information provided by these graphs it is necessary to describe the experiments in more detail. They include the construction of Omicron overlays and the construction of clusters

with peers assigned to Omicron roles. They have been performed using a laptop at 1.0 GHz speed and 384 MB memory. The employed JVM is JDK1.4. In summary, it can be derived that it takes less than a minute time and less than 200 MB of memory to construct a complex overlay like Omicron of size 100.000 active nodes. The observed linearity in terms of physical resources promises simulation experiments sized as large as half a million nodes in moderately more powerful machines dedicated for simulation. In order both to demonstrate the results and observe/debug the simulated experiments, graphical tools are necessary. A variety of graph tools can be used to draw static graphs. However, the graphical animation of the experiment provides additional information about the experiments, which can add great value. For this reason, the simulator can generate Nam-like traces that can be consumed by the Nam tool (Network animator). Since Nam has been designed to fulfil the needs of Ns-2 [FP01] and its implementation is based on C++ and Tcl languages, an alternative tool named Jarvis (Java Visualizer) [Gam00] proved to be more useful for the needs of the simulator.

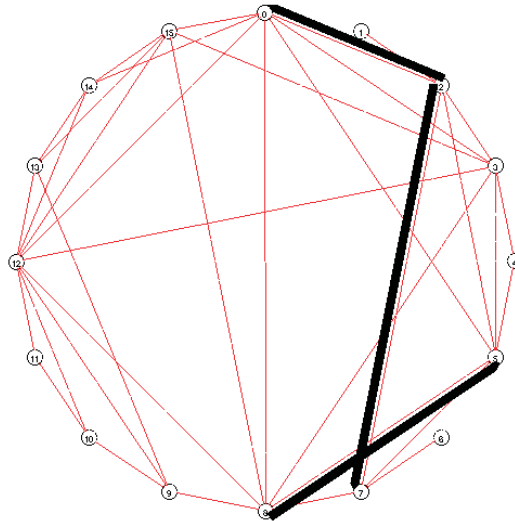


Figure E.2.: Jarvis screen-shot animating a Chord network construction.

Javis is capable of tracing Nam-files, but with more easy to understand software architecture. Javis has ben enhanced to fit to the needs of the P2P simulator in three ways:

- *Support directed links.* Javis and Nam support only bidirectional links.
- *Control overlay topology.* P2P overlay networks can be displayed easier.
- *Remove failed nodes and links.* Javis as well as the original Nam have no ability to remove failed nodes and links.

Figure E.2 shows a screen-shot of the enhanced Jarvis animating the creation of a Chord-based P2P overlay network.

Appendix F

Abbreviations

AGILE	Adaptive Group-of-Interest-based Lookup Engine
ACPH	Acyclic Continuous Phase-type
ADPH	Acyclic Discrete Phase-type
AoI	Area of Interest
APH	Acyclic Phase-type
AS	Autonomous Systems
BIFS	Binary Format for Scenes
C/S	Client/Server
CAN	Content Addressable Network
CCDF	complement cumulative distribution function
CDF	cumulative distribution function
CF	Canonical Form
CFS	Cooperative File System
COSMOS	Collaborative System based on MPEG-4 Objects and Streams
CPH	Continuous Phase-type
CPU	Central Processing Unit
CTMC	Continuous Time Markov Chain
CV	Coefficient of Variation
CVE	Collaborative Virtual Environments
DAMD	Distributed Algorithmic Mechanism Design
DDoS	Distributed Denial of Service
DMIF	Delivery Multimedia Integration Framework
DPH	Discrete Phase-type
DSL	Digital Subscriber Line
DTMC	Discrete Time Markov Chain
DFR	Decreasing Failure Rate
DHT	Distributed Hash Table
DNS	Domain Name System
EM	Expectation-Maximization
GB	Giga-Byte (2^{30} bytes)
GoI	Group of Interest
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
IC	Integrated Circuit
IFR	Increasing Failure Rate
IP	Internet Protocol

JDK	Java Development Toolkit
JXTA	JuXTApose
JVM	Java Virtual Machine
LFU	Least Frequently Used
LRU	Least Recently Used
LST	Laplace Stieljes Transform
MB	Mega-Byte (2^{20} bytes)
MGF	moment generating function
MIMD	Multiple Instruction Multiple Data
MMAPPS	Market Management of P2P Services
MMG	Massive Multiplayer Game
MPEG	Moving Pictures Expert Group
MRGP	Markov Regenerative Processes
MRL	Mean Residual Lifetime
MTTF	mean time to failure
NAT	Network Address Translation
NHMP	Non Homogeneous Markov Process
NPC	Non Player Character
NS	Network Simulator
ODRI	Optimal Diameter Routing Infrastructure
Omicron	Organized Maintenance, Indexing, Caching and Routing in Overlay Networks
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
pdf	probability density function
PH	Phase-type
PIER	P2P Information Exchange and Retrieval
pmf	probability mass function
RDP	Relative Delay Penalty
RPV	Rendezvous Peer View
SHARK	Symmetric Hierarchy Adaption for Routing of Keywords
SIMD	Single Instruction Multiple Data
SME	Small and Medium Enterprize
SMP	Semi-Markov Processes
SPN	Stochastic Petri-Nets
SRDI	Shared Resource Distributed Index
SRH	Symmetric Redundant Hierarchy
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VRML	Virtual Reality Modeling Language
WWW	World Wide Web

List of Figures

2.1. Backbone traffic measurements.	13
2.2. Evolution of event handling.	16
2.3. Directed de Bruijn(2,3) graph.	20
2.4. The dependability tree [ALR00].	25
2.5. Availability versus Reliability: Markov chain examples.	26
3.1. Overlay network design dimensions.	35
3.2. Chord architecture.	37
3.3. JXTA overlay network.	41
3.4. Brocade overlay network.	42
3.5. SHARK overlay network.	43
4.1. Two-tier overlay network topology.	57
4.2. Inter- and intra-cluster connectivity pattern.	58
4.3. Join sequence diagram.	59
4.4. Complete extension of a de Bruijn(2,2) digraph to a de Bruijn(2,3) digraph.	60
4.5. Extending a (2,2) de Bruijn graph, by adding nodes incrementally.	61
4.6. Removing unstable clusters.	64
4.7. Directed de Bruijn(2,3) graph without self-connected nodes.	65
5.1. Inter-role communication mechanism.	70
5.2. Lookup operation using cache indices.	81
5.3. Abstract description of the cache structure.	81
5.4. Lookup scenario steps.	85
5.5. Resource advertisement scenario steps.	86
5.6. Peer join scenario steps.	86
6.1. CCDF-based fitting with single distributions of the empirical curve.	96
6.2. CDF-based fitting with single distributions (head) of the empirical curve.	97
6.3. Peer life-cycle model.	98
6.4. Fitting of lognormal mixture distribution to empirical peer uptime.	100
6.5. Lognormal Mixture.	101
6.6. Lognormal Mixture Hazard Rate.	103
6.7. Burn-in cost.	107
6.8. Survivability comparison.	107
6.9. Cluster endurance distribution.	108

6.10. Expected time for endurable clusters ($E_{CW}(t_{fn}) > 0.99999$).	109
7.1. Functionality layers.	114
7.2. Component design pattern.	115
7.3. Core peer components.	116
7.4. Simulation engine.	117
7.5. Link modeling.	118
7.6. Euclidean distance abstraction.	119
7.7. Message handling.	121
8.1. Shortest path distribution.	125
8.2. Routing scalability.	126
8.3. Routing scalability: Omicron versus Chord.	127
8.4. Vicinity exploration: Omicron versus Chord.	128
8.5. Query routing load distribution.	130
8.6. Vicinity exploration versus load-balance.	131
8.7. Routing query distribution on de Bruijn graphs.	132
8.8. Routing workload on self-looped nodes.	133
8.9. Deterministic R-Shift algorithm.	135
8.10. Sampling distribution using random walks.	136
8.11. Standard deviation of sampling in de Bruijn networks.	137
8.12. Coefficient of variation of the cluster sampling mechanisms.	138
8.13. Cluster sampling inter-arrival distribution.	139
8.14. Cumulative resource popularity distribution.	141
8.15. Cache routing load as a percentage of the original Chord network.	142
8.16. Load distribution for replying queries.	142
C.1. Modeling Concepts in Virtual Environments.	182
D.1. Query routing load distribution on large networks.	183
D.2. Sampling distribution using random walks.	184
E.1. Simulator Performance.	185
E.2. Jarvis screen-shot animating a Chord network construction.	186

List of Tables

2.1. CVE content classification.	15
2.2. Classification of common overlay operations.	21
4.1. Requirements of Roles and servicing rules.	55
5.1. Common message structure.	69
5.2. <code>Accept</code> message structure.	71
5.3. <code>UpdateRoles</code> message structure.	71
5.4. <code>UpdateClusterMap</code> message structure.	72
5.5. <code>LookupRequest</code> message structure.	74
5.6. <code>LookupReply</code> message structure.	75
5.7. <code>PublishRequest</code> message structure.	75
5.8. <code>PublishConfirmation</code> message structure.	75
5.9. <code>SplitClusterRequest</code> message structure.	76
5.10. <code>SplitClusterConfirmation</code> message structure.	76
5.11. <code>MergeClusterRequest</code> message structure.	77
5.12. <code>MergeClusterConfirmation</code> message structure.	77
5.13. <code>UpdateIndexRequest</code> message structure.	78
5.14. <code>UpdateIndexConfirmation</code> message structure.	78
6.1. Parameters of fitted distributions.	97
6.2. Parameters of mixture distributions.	101
6.3. Peer up-times configuration example.	108
7.1. Simulator roles and overlay networks.	120
8.1. Sampling algorithms routing cost.	135
8.2. Mean value of cluster coverage sampling.	137
8.3. Exponential cluster sampling inter-arrival rate approximation.	140